ISSN 0956-9979 JANUARY 2000

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Under attach!** Consciously or unconsciously, this issue's features all centre around the growing dangers concealed in email attachments. When will we ever learn?

• **News and views:** Our extended news pages, starting on p.3, reflect the flurry of activity in the run-up to Christmas and the New Year. This month's Letters page is dedicated to Y2K predictions from AV's hall of fame.

• **What exactly is the WildList?** Shane Coursen gets back to basics in his full and detailed analysis of the functions and motives behind Joe Wells' famous list, starting on p.9.

• **Seriously Serial:** This month sees the first instalments of not one but two new series dealing with the topical issues of malware of the future and viral threats to *Lotus Notes*.

CONTENTS

COMMENT
In Support of Support 2

VIRUS PREVALENCE TABLE 3

NEWS
1. A Virus of Biblical Proportions? 3
2. Pleading Guilty 3
3. Showing Executables the Exit? 3
4. Double Whammy 4
5. Silly Season's Greetings 4

LETTERS 5

VIRUS ANALYSIS
Don't Press F1! 7

FEATURE
Mythconceptions 9

OPINION
Learning from Experience 14

A DAY IN THE LIFE
Viruses: A Way of Life 16

FEATURE SERIES
1. Malware Do You Want To Go Today? Part 1 18
2. Lotus Notes and Email Risks – Part 1 20

TUTORIAL
The Final Frontier? 22

END NOTES AND NEWS 24

COMMENT

In Support of Support

“ … there is a real difference in the quality of service from one AV vendor to another. ”

A new year, a new decade, a new century, and even a new millennium (pedants can shut up – who wants to ruin a party?). Congratulations everybody, we've survived another decade without viruses taking over the world. Anti-virus scanners are still the preferred defence for most companies, despite the predictions of some that they wouldn't be able to keep up with the increase in virus numbers, and they still seem to be doing a reasonable job of stopping viruses from spreading.

Anti-virus software has definitely improved. Just a few years ago there was a clear difference in the detection rates of some products – curiously, European products were typically better at detecting viruses than their American cousins. Today, the major players' software is pretty similar in terms of detection rates (occasionally there is a 1% difference here, a 2% difference there, but nothing significant). Long gone are the days when the European AVs could consistently detect 20% more viruses than their American counterparts.

Virus Bulletin can take some of the credit for this improved software quality. Their competent, regular tests of detection rates have helped anti-virus companies realise that detection of viruses is fundamentally more important in an anti-virus product than pretty user interfaces or bells and whistles. After all, an anti-virus product that cannot detect viruses is as useless as a word processor that cannot process words. Whilst other magazines have shied away from testing anti-virus products' core function, *VB* has focused on the most important aspect: can it do the job?

The market has changed. Detection rate has become less of a differentiator for corporations choosing an anti-virus solution because most of the major products are doing a pretty similar job in that regard. So, how do corporations decide which is the best anti-virus product today? At the *Virus Bulletin* 1999 conference in Vancouver David Phillips of *The Open University* gave a talk entitled ‘Who Cares Wins?’ about his organization's experiences with the customer service of different anti-virus companies. It reminded me of an issue I have been thinking of a great deal lately.

Have you tried ringing your anti-virus vendor recently? Sent them an email asking for technical assistance? What was the response? Did they respond at all? If you work for an anti-virus developer, have you played ‘mystery shopper’ and tested your own technical support and customer service hotlines? It seems to me that the AV marketplace is shifting. Whereas detection technology differentiated which products a corporation should purchase five years ago, today customer service and technical support have experienced a dramatic rise in importance. It appears from customers I have spoken to that there is a real difference in the quality of service from one AV vendor to another. The difference becomes clear when considering how well the vendor looks after you when you do experience problems – do they answer the phone promptly, are they knowledgeable when you speak to them?

Anti-virus software is complex, and it is inevitable that sometimes things will go wrong (yes, even in the product I represent!). For this reason I think it is essential to consider how well the company reacts when you do have a problem. Don't get me wrong – high detection rates are still vital in an anti-virus product. It's just that customer service and support (which includes the timeliness and quality of updates) has risen in significance for the corporate purchaser. Corporations should evaluate the quality of support they are receiving from their anti-virus vendor, and bear in mind these issues when considering which products to switch to.

The costs of supporting your chosen anti-virus product inside your organization can often outweigh the initial purchase price. It's important for corporations to consider the cost of *owning* the software as opposed to the cost of *buying* it in the first place. When you buy an anti-virus product you should be purchasing not just the software but the technical support and customer service that comes with it.

Graham Cluley, Sophos Plc

NEWS

A Virus of Biblical Proportions?

Win95/Babylonia, posted on 3 December to a 'cracks' newsgroup, represents a chilling new direction for network-aware malware. Babylonia patches WSOCK32.DLL in a similar manner to Win32/Ska, allowing the virus to monitor outgoing email messages and add an infected executable as an attachment. The virus also distributes itself as 2KBug-MircFix.EXE via *mIRC* scripting tricks. Of most interest, however, is its innate ability to download 'plug-ins' to enhance its capabilities.

Babylonia infects PE EXE and *Windows* Help (HLP) files. It is limited to *Windows 9x* platforms because of its dependence on VxD interfaces, hooking the IFS chain and elevating its code to run at Ring0 much like Win95/CIH.

At the time of its discovery, four plug-ins were available at its 'upgrade site'. One implements the *mIRC* distribution mechanism mentioned above, while another adds 'greetings' comments to AUTOEXEC.BAT after 14 January. The third sends an email message to a Hotmail address as an infection counter, and the fourth plug-in drops and runs an infected EXE file. Pressure on the ISP hosting the 'upgrade site' resulted in that site being closed. *VB* plans to feature a full analysis in the near future ▮

Pleading Guilty

On Thursday 9 December, David L. Smith, the author of Melissa, pleaded guilty to a second-degree charge of computer theft. Appearing in Monmouth County Superior Court (US), Smith admitted to the charges but claimed that he did not anticipate the scale of the problems that Melissa would cause. According to *ZDNet*, companies such as *Microsoft*, *Intel* and *Lockheed Martin* were forced to shut down their email gateways during Melissa's rampage.

Smith now faces a second hearing in February of next year, when he will receive his sentence. He could be fined up to $400,000, and/or jailed for up to 10 years. Next month, Sarah Gordon reflects on the impact and implications of Smith's conviction ▮

Showing Executables the Exit?

Throughout 1999 we have seen viruses spread rapidly even when user intervention, such as running an email attachment, is required. Lessons should have been learnt by all. However, recent events suggest this is not the case. Lessons have not been learnt by users, both corporate and home. Not even by many System Administrators, it would seem.

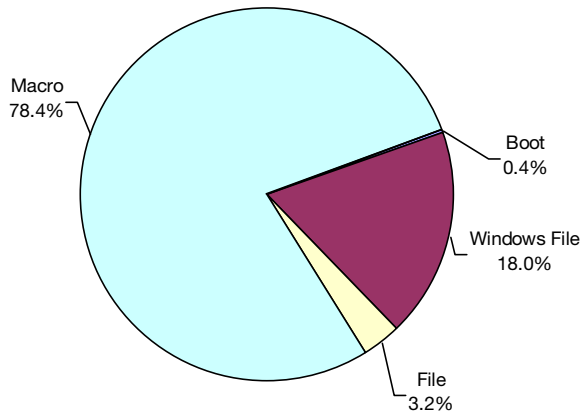
Recent weeks have seen two Christmas games circulating wildly on the Internet – 'Elfbowl' and 'Frogapult', both written and released by *NVision Design Inc*. Unfortunately,

Prevalence Table – November 1999

Virus	Type	Incidents	Reports
ColdApe	Macro	960	46.3%
Win32/Ska	File	184	8.9%
Marker	Macro	124	6.0%
Laroux	Macro	106	5.1%
Win32/Pretty	File	101	4.9%
Melissa	Macro	85	4.1%
Ethan	Macro	84	4.1%
Tristate	Macro	68	3.3%
Freelinks	Script	62	3.0%
Class	Macro	61	2.9%
Win32/ExploreZip	File	31	1.5%
Bablas	Macro	20	1.0%
Story	Macro	17	0.8%
Win95/CIH	File	14	0.7%
Cap	Macro	12	0.6%
Win95/Babylonia	File	12	0.6%
Thus	Macro	11	0.5%
Opey	Macro	10	0.5%
Win32/Fix	File	10	0.5%
JB	Macro	9	0.4%
Win32/Kriz	File	9	0.4%
Win32/Mypics	File	8	0.4%
Prilissa	Macro	7	0.3%
Others [1]		69	3.3%
Total		2074	100%

[1] The Prevalence Table includes a total of 69 reports across 27 other viruses. A complete summary can be found at http://www.virusbtn.com/Prevalence/.

Distribution of virus types in reports



but rather predictably, hoax messages warning of their viral content (set to trigger on 25 December) circulated equally widely soon after.

And the result? Hoards of home users are concerned about their machines – concern born primarily out of considerations of finance and inconvenience, one would suspect. Also, hoards of corporate users are worried about what damage they may have caused their office machines, and the implications this may have upon their next review!

The underlying strategy behind these games is quite simply multimedia marketing. *NVision Design Inc* advertise their services as aggressive marketing solutions. Their latest brainchild appears to be a project dubbed *NStorm*, which according to their Web site:

> *"… is a patent-pending process that combines games, marketing messages, and people's compulsion to share amusing emails to drive millions of hits to our clients web sites."*

In this instance, the hits on the *NStorm* and *NVision Design Inc* sites will no doubt have increased. Hits on the *VB* and AV product developers' sites have also increased thanks to the hysteria that has followed.

As it happens, the originally released games were harmless. The only slightly dubious activity occurs when Elfbowl is first executed – it attempts to connect to the *NStorm* Web site, enabling them to keep a tally count of 'Elfbowlers'. The problem with this form of direct marketing is the confusion it causes users. A year on from the initial release of Win32/Ska, *VB* still sees countless users each week infecting their machine by running the emailed HAPPY99.EXE. With the waters of safe computing further muddied by this latest saga, perhaps it is time for more organizations to block the transmission of executable program files for ordinary users ▮

Double Whammy

The November issue of Hungarian computer magazine *Uj Alaplap* sent out a CD infected with both W97M/Class.B and W97M/Opey.A. These two viruses can propagate together as a double infection. Class.B infects the global template when an infected document is opened and then, using the AutoClose macro, infects all documents as they are closed. During infection it clears the class module's contents. Opey.A infects the global template and documents on opening, closing, saving and printing. It uses its own A_OPEY_03 module for storing the code. While infecting the virus deletes all VBA modules except its own and the class module (which would obviously cause serious problems for *Word*).

Opey.A triggers on almost any action, Class.B only on document closing. Uninfected, open documents will sooner or later be closed. Then Opey.A activates, thanks to its FileClose macro. Since, during a normal document close, the FileClose macro activates before AutoClose, Class.B gets a second chance to infect. This causes a minor interaction between the viruses. Class.B's weak polymorphic engine inserts a comment line to every second code line that combines user name, system date, printer info and document name. Opey.A's payload changes the *Word*-registered user name to OPEY.A – the name in all infected documents. There is practically no chance for either virus to break out of the double infection. Opey.A can, but only if the document is closed with automacros disabled.

If an uninfected document (Opey.A's AutoOpen macro means this can only happen if it was opened with the SHIFT key pressed) is closed using the CTRL+F4 key combination, the FileClose macro is not executed. Thus, only Class.B's AutoClose gets called and only this virus will infect the document. However, as the user name is still 'OPEY A.', the document will display signs of the doubly infected global template ▮

Silly Season's Greetings

Virus Bulletin receives many a press release which stretches the truth somewhat. Our product reviews often make mention of the ludicrous claims featured on product packaging. *VB* staff were amused to read a press release this month which was headed '*Content Technologies* eradicates all macro viruses'. Wonderful news for all! Falling headlong into the hyberbole trap, the company went on make a woefully ambitious new year's resolution by promising:

> "*E-mail users are no longer at risk from macro viruses embedded in Microsoft Word documents.*"

Rife with biological analogies, *Content Technologies* claims to have discovered a 'macro gene', thereby enabling 'all new and current macro viruses to be identified by genetic signature with 100% accuracy'.

This 'new technology' is being released by *Content Technologies* to protect customers from email-borne threats over the millennium period. It is available as a plug-in, dubbed *Y2K Scenario*, for their *MAILsweeper* product, and enables the macro content of *Word* document attachments to be stripped (if so desired by the adminstrator).

More stringent controls at the gateway (e.g. stripping macro content) are welcomed – the majority of users do not need to exchange macro content within documents at all. However, dressing this technology up as a Y2K issue is not helpful, and merely contributes to the overall media hysteria of which we are all rather bored ▮

LETTERS

'This is the forecast… '

[*Unless you've been stockpiling provisions behind the locked doors of a Y2K-resistant bunker for the last ten years you'll be familiar with most if not all of the names on this month's Letters page.* Virus Bulletin *asked high profile anti-virus experts to give their predictions for 2000 and bid farewell to the 1990s in 140 words or less.*

As you can see, the results may be pessimistic, but solutions are already under development. Ed.]

Jimmy Kuo – NAI

I'd like to offer two thoughts, one for the present and one for the future. Is there any reason why anyone should be sending executables any more? Today, if I ever need an executable, it's likely to be a full setup package! If you have an email server, bounce all .EXEs with an error message which says, 'If you intended to send that executable, send it in a passworded ZIP file!' That should cut down on all these EXE Worms. And then, if the sender didn't know he was infected, he surely will!

Get your head round the 'No-Click attack!' (as embodied right now in BubbleBoy – the use of exploits to hit you just because you're browsing a Web page, or reading your email, for instance). If you don't, the hackers will do it for you. This is our future.

Pavel Baudis – Alwil

There used to be one simple operating system and new viruses appeared very rarely. Today, there are many new operating systems, networks, new applications with new data formats, new technologies, tools and features. However, the most critical aspect today is the speed of spreading. Viruses, Worms and other malicious programs are now able to spread so quickly that it is quite difficult for both users and AV vendors to react in time and in the right way. If the interval between such incidents gets any shorter, users could find it simply unacceptable.

Now is the time to change anti-virus technology to more general and less virus-specific methods. It will soon be time for application developers to finally change their minds and implement fewer features and more security aspects in their products – otherwise their software may be found unusable by users.

Sarah Gordon – IBM Research

Rapidly changing technologies create new worlds for viral replication; virus problems evolve as these shifting sands of technology push us into a world where viruses are everyday occurrences. The basic details – 'how does it replicate?' and 'what should we call it?' – will become increasingly irrelevant, as users experience viral ubiquity, the result of continuing trends in network-aware code. The solution? Continued research and development of technologies which find, cure and immunize huge sections of cyberspace against cyber-threats – faster than threats can spread.

There's another, basic aspect here: the humans involved, and how we develop virtual communities. What is acceptable and what is not? We must continue to address these important international issues from cultural and legal as well as technological perspectives.

The virus problem is not, never was, just about technology: people are the crux of both problem and solution.

Eugene Kaspersky – Kaspersky Lab

Difficult problems appear more often and solutions must be developed quicker. The most advanced DOS virus is no more complex than the average *Windows* one now virus writers 'research' *Windows* and pass on their experience. A significant problem will be the improvement of macro and script viruses. New infection methods mean reconstructed scanners; viruses for new platforms mean spending more on research. Internet Worms do not use security breaches, yet. But they might. Use the strictest Internet security settings.

We'll have to pay more attention to object embedding, too. Information is constantly created, transmitted, stored, and restored. It is in hundreds of formats which we must open and scan for possible viruses, even if they can't exist there. What about new hardware? Imagine a mobile phone that can be programmed via a standard call connection. 'Who's watching your microwave oven?'

Fridrik Skulason – FRISK

The challenges the anti-virus industry faces will not be solved or disappear quietly after 1 January 2000 – if anything, the situation will get worse.

I am not overly concerned with the possibility of a flood of viruses or Worms on (or around) 1 January 2000, although I believe some virus authors may have been 'saving' their recent creations for release around that time. What concerns me most is the ever-increasing variety of new platforms capable of supporting virus development, and the increased complexity those platforms offer.

We are rapidly reaching the point where it is getting impossible for any single individual to be an expert on all the possibilities of virus development. That is a scary thought for anyone who has been active in anti-virus development for over 10 years.

Nick FitzGerald – Independent

What will the new millennium bring us in virus and anti-virus technology? Not wishing to appear to be flogging a dead horse (see last month's Opinion article), but scanning is dead. Corporate users should take responsibility for the code running in their LANs and demand better tools for doing so.

The good news is, much existing virus detection technology could easily be converted into true integrity management technology – file system filters, file 'finger-printing', smart file-typing and decomposition etc, would all figure in such a system.

The bad news – we will not see it unless the vendors see a market for it. Thus, it is time for the users to decide that there must be a better way. There is. Be part of the movement to break the tyranny of the virus scanner – get on your vendor's case today!

Jakub Kaminski – Computer Associates

There is one thing that history teaches us over and over again; it is that history teaches us nothing. Old hoaxes, and particularly the second coming of ExploreZip are the best examples of this.

What will happen a second after 23.59.59 on 31 December 1999? Any semi-intelligent guess based on what we've seen so far seems as good as others. It is easier to predict what will not happen. How? Browse the Internet looking for the hype on 'Y2K viruses'. Read it and forget about it. Pity that a significant amount of that rubbish could be found on the anti-virus sites.

There will be two reasons for problems: a) missed Year 2000 incompatibilities; b) malicious human activities. Targeted attacks (hacks and malware released in selected areas) are the real threats, not the invisible viruses out there waiting to explode.

Costin Raiu – GeCAD

In my country, the pronunciation of the zeroes in 2000 has a meaning similar to the word 'trouble'. Speaking of trouble, I see a new custom in the software business – to name software products 'YeGreatSoft 2000'. But how many of them are worse than the previous version of the same product? Indeed, far too many.

Radical problems require radical solutions. Let the weak die, let the stronger survive. The choice is yours. If you read this – well, wise choice. To all the others, better luck next time round.

And let's not forget the mistakes of the past – in this small world, everything is getting larger each day. Will we handle 200,000 viruses sometime? Will we handle them in only one day? Take a moment and think about it. Happy New Year everyone!

Carl Bretteville – Norman AS

Networked computers take us back to something that resembles a mainframe with more computing power and improved reliability. Distributed computing is not new, but has not been widely used until this year. People use their own computers to form a huge cluster. The drawback? The distribution of work is manual. Network server clusters will automate the distribution of work to available resources. They will have distributed file systems. The AV industry will have to tackle this soon.

The idea of containing information in a single, isolated 'bubble' may not be true any more. Add to this the success of scripting languages that will run in applications on your desktop and network. The decade I have worked in this industry has been far from dull. The next will be more challenging. The real trick will be to keep it fun.

Randy Abrams – Microsoft

I tend to shy away from making prognostications about anti-virus. I'm an adept user, but not a researcher.

As I was preparing for a training presentation I gave yesterday, and having seen dozens of emails asking if ELFBOWL.EXE was really going to wipe out every hard drive on 25 December, it did occur to me though that it is likely that we will have to see a shift in how we (computer users) go about getting software.

People are going to have to stop sending programs (especially joke programs) to each other and start sending Web addresses, or perhaps ftp addresses. Even in corporations I believe policy will mandate that locations of files are sent rather than the files themselves.

The analogy I used in the training presentation was that getting a program in email from a friend, instead of downloading from the developer's Web site, is the computer equivalent of sharing a needle!

Carlos Ardanza – Panda

1999 has undoubtedly been characterized by the use of email as a means of massive virus infection. If we have learned anything this year, it is that it is no longer enough to update your anti-virus software every three months, but rather this must be done *at least* daily.

For the year 2000, this trend is expected to increase considerably. Pessimism aside, we'll just have to get used to the idea of tackling viruses with the execution capacity of BubbleBoy, and the spreading and destructive payload of ExploreZip and its variants. It will be the year of script viruses hidden in HTML format emails, and the power of *Windows Scripting Host* and HTML applications.

As for major corporations and anti-virus companies, it will be the year of specially designed email anti-viruses and instant responses to emergency situations.

VIRUS ANALYSIS

Don't Press F1!

Eugene Kaspersky
Kaspersky Lab, Russia

In my opinion, Win95/SK, discovered in March 1999, must have been one of last year's most significant viruses. Most importantly, anti-virus scanners can see practically no sign of it. The entry point address and the code at the entry point are not modified, and there is nothing in either the file header or structure to point to virus code. There are only two ways of detecting it: scanning the entire Code section of the file (often sizeable) for a virus entry routine, or scanning and x-raying the whole Fixup section. These methods definitely slow down the scanner's speed and present anti-virus developers with a dilemma: scan speed or detection quality.

SK is a slow infector: before infecting it checks many conditions and affects very few files – about ten EXEs in a standard *Windows95/98* installation. The same is true of Help (HLP) files and archives. The virus delays its infection routine for one minute before the first infection, and infects HLP files and archives only if there has been no access to these files for the duration of two minutes. All that makes virus analysis hard – more time is spent getting virus replicants than understanding virus structure.

In my opinion, this virus is very dangerous. When disk files are accessed, it checks their names and if it finds several anti-virus programs (ADINF, AVPI, AVP, VBA, DRWEB) deletes all the files it can from all directories on all disks from the C: drive to the Z: drive and then halts the system with the Fatal_Error_Handler VMM call. For some reason the COMMAND.PIF file is also deleted each time SK installs itself. It does have bugs which are lethal under some *Windows95/98* configurations, resulting in 'blue screens' with the standard 'General protection fault' messages. While installing, the virus, depending on its random counter (one in 48 times) displays the message:

```
<C> 1997 VBA Ltd. Email:support@vba.minsk.by
```

Installing Memory Resident

SK's 'resident' copy works at the VxD (Ring0 – *Windows* kernel) level. DOS programs and *Windows* applications cannot access this area by standard methods, so the virus uses several tricks to install its code there. When the DOS dropper is executed, the virus checks that *Windows* is installed and returns to DOS if it is not. Otherwise, the virus uses DMPI calls to get access to Local Description Tables, patches them, and switches its DOS 16-bit code to protected mode 32-bit. SK's virus code then works as a part of the *Windows* kernel and can access all the necessary *Windows* functions.

When an infected PE file is run, SK also patches the system memory allocation tables and switches its code from application level (Ring3) to kernel level (Ring0). It then passes control to the same installation routine as infected DOS droppers do. The installation routine, when it takes control, allocates a block of system memory, copies the virus there, hooks *Windows* functions and releases control. The original virus copy either returns to DOS or restores the host PE file data and code and returns.

Virus Hookers

Win95/SK's first hook is a callback procedure for I/O port trapping. It hooks port 534Bh and uses it in its 'Are you there?' call. Reading from this port under infected system conditions returns 21h ('!') in the AL register. Both DOS and *Windows* virus installation routines try it before installing memory resident. The number of this port was used to name the virus – 534Bh is 'SK' in ASCII. The second 'standard' hook for resident *Windows* infectors is the File System API hook. It intercepts file opening, renaming and file attribute get/set functions. On these calls the virus runs its infection routines.

The third hook intercepts the InstallFileSystemApiHook function itself (the one used to install the previous hook). This call is used by SK to hide its hooker: when a new hook is installed, *Windows* calls this function. Then the virus intercepts it, removes its own IFS API hook, installs a new one, and re-installs its virus hook. Thus, the virus hooker is always top of the hook list and the first to receive control when disk files are accessed.

There is another trick here. To access file system drivers SK uses the address of a 'native' system handler. It gets this address from the purely documented field in the caller's stack, and uses it to perform direct calls to necessary file functions. Together with the 'always-on-top' trick this is used to avoid AV detection: SK's calls bypass anti-virus monitors and it is impossible to catch its IFS API hooker.

Infecting Files

When the IFS API hooker gets control and the infection routine is activated for the first time, it infects the *Windows* shell application (usually EXPLORER.EXE). To locate it SK opens the SYSTEM.INI file in the *Windows* directory, scans it for the 'shell=' instruction and infects it.

The virus cannot modify this file, so it uses an 'upgrading' trick. It copies the file with another name (changing one letter on the file name: EXPLORER.EXF, for instance), infects the new file and forces *Windows* to 'upgrade' the original file with the infected one. This is done in the standard way.

If the *Windows* shell is already infected, the virus infects the file that is being accessed. First of all it checks the file name extension. In EXE, SCR and DLL files the virus jumps to the PE file infecting routine; it patches HLP files with its dropping code; RAR, ZIP, ARJ and HA archives have SK's executable dropper inserted into their contents.

Infecting Windows PE Files

While infecting PE files SK encrypts and writes its code to the Relocation (Fixup) section and overwrites the data. In most cases all applications are loaded to the same addresses, and no relocations are used. Rarely, when an application is loaded by another host application to that host's memory, the *Windows* loader processes the Relocation Table and makes necessary changes to the application's code. To fix this SK modifies the PE header fields, clears Relocation flags in the Characteristics field and sets the address of the Relocation section to zero. The affected application cannot be loaded at another address but works perfectly as a standalone one.

SK then makes necessary changes to the Relocation section header, setting its characteristics as Readable and Initialized and, depending on its random counter (1 time in 8), changes the section name to a random one, encrypts its body and writes it there. The virus also checks the Relocation section size – is it big enough for virus code? – and infects files only if the virus code fills in the section body. Thus, SK does not increase PE files' size while infecting them.

The virus does not modify the entry address to get control when infected programs run. It patches randomly selected code in the program body and writes the JMP_Virus routine to there. So the virus code is not executed immediately the infected program is activated, but only if a patched branch gets control. In some cases, when this branch is executed very rarely (an error handling routine, for instance) the virus 'sleeps' in the program for ages and jumps out later.

To locate a position to patch, the virus scans blocks of the victim file for C/Pascal subroutines. These routines have standard headers and footers, and SK just needs to find them to see that there is enough space for its JMP_Virus code between the subroutine entry and exit instructions. This is not just a single JMP instruction. It is more sophisticated – a polymorphic loop that decrypts the main virus code with up to 168 bytes of code before passing control to it. In some cases (when the *Windows* shell is infected, or no C/Pascal routine is found) the virus writes its JMP_Virus patch directly to the program's Entry address. Then its code gets control immediately the infected file is run.

The SK virus does not mark infected PE files in any special way. PE files cannot be reinfected and SK will not touch those without Relocation sections. Having said that, it does clear its address in the PE header while infecting. The one exception to this is that while it is infecting the *Windows* shell (EXPLORER.EXE) the virus writes its ID stamp (21h) to the DOS header of the CRC field.

Infecting Archives

While infecting four types of archive (see above) SK parses their internal formats, adds a DOS dropper to the end of the archive and carefully modifies the structure, including CRC fields. Only archives that have no fewer than ten files and at least one COM or EXE file are infected. It adds a DOS COM file that has a randomly generated name and COM or EXE extension. The virus has one more branch for infecting RAR archives compared to other routines. This branch looks for 'stored' files in the RAR archives, and if a file has a specific date and time stamp, SK reads and executes stored data. This trick allows the virus to run other programs without user intervention – it can 'upgrade' the virus, run a spy or any other kind of program.

Infecting Help

Windows Help files have their own script sub-programs (macros). These are automatically executed when WinHelp activates Help files, and the script language is powerful enough to access disk files, create and execute them. The virus uses this feature to infect these files. It writes its DOS dropper to the end of the HLP file as an 'overlay', modifies the internal HLP structure and adds its own script. This script has ten instructions written to the [CONFIG] section which is automatically processed when an infected Help file is activated. The instructions extract and execute a small (about 380 bytes) virus loader. The virus script creates this loader as a DOS file (with a random name) on the C: drive, writes a code there and executes it. Then the virus loader takes control.

The virus loader's code in HLP script is converted to ASCII 7-bit format. When this code takes control, it converts itself back to binary code, then gets the name of the host HLP file, reads the header, gets the offset of the virus 'overlay' code, reads it, overwrites its own COM file and re-executes it. Then the DOS dropper gets control, and SK installs itself into memory… A final note about this potentially destructive virus – while dropping its COM file the virus script checks the presence of the *NT*-specific file C:\\NTLDR and returns if this file is found. Be on your guard against SK.

Win95/SK

Aliases:	None known.
Type:	Memory-resident, parasitic, polymorphic *Windows 9x* infector.
Infects:	PE files, *Windows* HLP files, inserts droppers to archives of several types.
Self-recognition:	PE files cannot be infected twice. Uses port hook to detect its TSR copy.
Removal:	Delete infected files and restore from backups.

FEATURE

Mythconceptions

Shane Coursen
Network Associates Inc

The WildList is known and used by nearly all major anti-virus product reviewers, anti-virus certification bodies and anti-virus software vendors. It is generally regarded as *the* source of information when referring to viruses found with prevalence. Its success is not surprising in view of the fact that its reporting 'Participants' are in the best position to measure the computer virus threat accurately.

The purpose of this article is to offer readers clarification of a few of the most common misconceptions regarding The WildList. Each section lists one misconception. Following each section's heading is a description (in italics) of the misconception. The description exists so the reader gains, from the perspective of *WildList Organization International* (*WLO*), an understanding of the misconception. Since *WLO* recognizes each of the following as a misconception, it will attempt to give an accurate and conclusive description of its true methodologies and procedures. Lastly, solutions, if necessary and wherever possible, are outlined.

Background

Unless you have had your head buried in the sand for the past six years, you will already know about a product called The WildList. You have probably also heard about the co-operative known as *WildList Organization International*. Nonetheless, here is a very quick overview.

The WildList consists of collated data regarding real-life computer virus incidents as received by 55 anti-virus researchers and two corporate reporting groups. People who report virus incidents to *WLO* are known as Participants. Participants are typically paid professionals in the field of anti-virus research. Together, *WLO* Participants represent every major anti-virus software package available. *WLO* Participants are located in over 40 countries. Their reports represent real-life computer virus incidents occurring in over 80 countries.

Managing the collection and collation of data reported by the Participants is a team of five. This team also works directly in the anti-virus industry and is known as The Board. The Board is advised by several leading anti-virus industry figures, known as Advisors to The Board. In sum, 60+ people form the volunteer-driven co-operative known as *WildList Organization International*.

Misconceptions

'Not all viruses found on users' computers are listed on The WildList.'

Part 1 – The most common statement and question combination put to *WLO* from the user population is 'I found this virus on my computer, but it isn't listed on The WildList. Why not? Because of this, you can't tell me The WildList reflects reality!'.

The trouble lies in the user's understanding of what The WildList is supposed to represent. If this were an actual problem, the obvious solution would be to change the underlying design of The WildList. Instead of implementing a minimum reporting guideline, *WLO* could ask its Participants to report every virus incident they hear about – regardless of whether it was backed up with a physical sample. In addition, *WLO* could ask its Participants to report virus incidents originating from technical support calls, even those where the caller was only asking about a particular virus because they think they had an infection. Lastly, instead of separating viruses reported by one Participant from those reported by two or more, *WLO* could list every single virus name in one giant WildList.

In four words, this will not happen. As described later, doing so would severely dilute the importance of The WildList. Not listing all viruses ever found is not a problem. Believing that it should is a misunderstanding of what The WildList is supposed to represent. In its current definition, The WildList 'works as designed'.

What is meant by 'as designed'? In order to describe The WildList – especially those facets that may be a little too abstract for people to be *au fait* with automatically – there are a few terms that first need to be defined. While some of the following terms may be familiar as they are often used generically throughout the anti-virus industry, the following definitions are given here specifically for the purpose of this article.

The WildList: A list of computer viruses (and more recently, other virus-like threats) verified to be found spreading throughout diverse user populations worldwide. In this article, this term will take two forms – depending on context. In one form it will refer to The WildList – the complete list. In its second form, it will refer *only* to the top half of the list – *the* WildList – that which is used in anti-virus software certifications.

In the Wild: The most-used term in the anti-virus industry. It references a virus that is found on a user's computer system – one whose operator did not want the virus on his system to begin with. For the purpose of this article In the Wild refers specifically to a virus that is listed in *the* WildList. It is sometimes shortened to ItW.

In the Field: A virus that is found on a user's computer, but not listed on The WildList. I use this term because I talk with people who have different interests. As an employee of

Network Associates Inc I am often told about viruses 'found in the wild'. As collator of The WildList, I receive samples of viruses found In the Wild – those that are intended for inclusion on The WildList. Using the term 'In the Field' helps me to differentiate between those viruses that are 'out there' (found on users' computer systems), but not listed on The WildList.

Supplemental List: Viruses listed on the bottom half of The WildList. While presented for inclusion on The WildList, these viruses are not yet officially In the Wild. As the old saying goes, it takes two. In this case, it takes two Participants reporting the same virus for it to be officially termed In the Wild. Two reporters reporting the same virus (that has occurred in two separate incidents) is the required threshold that must be met before detecting and repairing the virus is made mandatory for all anti-virus products. This two-count threshold allows *WLO* and anti-virus vendors to focus on a manageable and specific set of viruses that are known to have sustained prevalence ItW.

Now that some basic terms have been defined, it is time to describe 'as designed'. I will try to keep this short and simple. The WildList is designed to list those viruses and virus-like programs that are positively identified as spreading throughout diverse user populations.

It is a list the anti-virus vendors agree as important for *all* anti-virus software programs to detect and repair. It is a tool created and maintained by the anti-virus industry, and more recently by corporate entities relying on anti-virus software for their first line of defence. The WildList benefits the general user population, anti-virus vendors, anti-virus reviewers and anti-virus software certification bodies. The WildList is, of sorts, the world's most wanted list.

'Not all viruses found on users' computers are listed on The WildList.'

Part II – The most common statement and question combination put to *WLO* from the anti-virus research side is 'This virus was reported to me by my customers, but it isn't listed on the WildList. Why not?'.

There are several reasons why a virus – as seen by an anti-virus vendor – may not show up on The WildList. The most common follow:

It is possible that the virus *is* in fact on The WildList, although under a name different from that which was reported by the Participant. (For more information on why a virus might appear on The WildList by a name different from the reporter's, see the section under the subheading *Names listed on The WildList do not match the names used by anti-virus products*).

The WildList can list only those viruses reported by its Participants. Vendors, while they may have a representative Participant, do not report to *WLO* themselves. Vendors should encourage the Participant who represents their company to report all viruses meeting the minimum criteria to *WLO*. Vendors who do not have representing Participants – but who wish to – should contact *WLO* for further information on this.

Participants should not be shy about reporting a virus immediately – even if their product does not currently detect that virus. To withhold such information could only allow the virus to spread further. Knowingly allowing a virus to spread may result in reduced respect for the anti-virus industry.

Of considerably more impact is the reality that withholding reports only results in a greater amount of firefighting in the future (of the virus, which could have initially been made a high profile target. Find references to 'high profile target' in the sections *Only problem viruses are reported* part II and *The WildList lists only new viruses*).

Only the reports that come through proper channels are recorded properly. The Participant may simply have not reported the virus or may have sent the report in a non-standard fashion. Participants are requested to abide by the report form instructions.

Each and every virus reported to *WLO* must meet strict minimum criteria. The minimum reporting criteria are as follows. Most importantly, the Participant must take into consideration where the report comes from. If the report comes from an individual user, then the Participant must have verified a minimum of two reports over the past one month period.

If the minimum two-report criterion is met, the virus becomes reportable to *WLO*. If a single report originates from a corporate entity, the minimum is automatically met, and the virus becomes reportable to *WLO*.

The WildList was originally designed to be a list of viruses the anti-virus vendors agree as important for *all* anti-virus software programs to detect and repair. While a user negatively affected by a virus would disagree, a virus seen only once by one user and reported by only one Participant does not meet the agreed upon industry-wide 'important-to-detect-and-repair' criterion. In such a case, the virus may not appear on The WildList at all, or may only appear on the supplemental list.

In all cases, the minimum criteria require the incident to be verified with a physical sample of the virus. For example, if the Participant were a Technical Support Representative, two telephone calls referencing the same virus would not make for a viable report. A sample must exist in conjunction with the report.

While not very common any more, Participants sometimes send in reports without accompanying samples. In strict accordance with *WLO* practices, any virus name reported that is not accompanied by a corresponding sample cannot be considered for inclusion on The WildList. Furthermore, the submitted sample must eventually be proven viable (i.e. that it will replicate).

Cross-reporting

In this case, cross-reporting is 'to allow reports originating from the same incident to be counted twice – each report counted as a unique incident'. Cross-reporting, if allowed, would result in viruses appearing on *the* WildList when they should appear only on the supplemental list. In turn, this would result in anti-virus vendors being tested against a possibly inconsequential virus.

To make sure this does not happen, a check is made against cross-reporting. It has always been the practice to look for, and weed out, possible duplicate reports. As such, while worth mentioning, this is pretty much a non-issue.

The solution all along has been to rely on the Participant to fill in their report form properly. In addition to reporting the virus name and its frequency, there is also a place for the Participant to report the country of origin. Country of origin is where the virus incident occurred. When the Participant report forms arrive, any forms containing the same virus name from the same country within the same one month timeframe become suspect duplicate reports.

When they come across such suspect duplicate reports, the collators of The WildList contact the reporting Participant. The Participant is asked to supply further, more detailed information. The information requested is the name of the city where the incident occurred. If both Participants respond with the same city name, more and more detailed information may be obtained right down to asking about the type of business affected and, if necessary, the name of the business itself. The business name is known only to the collators of The WildList and is, of course, never published.

Along with the query for city/location information, the Participants are also asked to confirm their belief the virus *is* In the Wild. While somewhat subjective, this is an important question to ask. It is a good indicator of how strongly the Participant feels about their report. It also allows the Participant to make last-minute updates to their previously submitted report form. There have been several cases where Participants have witnessed further incidents of the virus between the time they sent in their original report to the time when they are being asked to supply more location information.

'Only problem viruses are reported.'

Part I – Viruses detected and repaired by scanners without encountering problems are rarely reported. Only those viruses that are problematic for scanners to detect and/or repair are reported to vendors. In turn, Participants who work for the vendors do not report those viruses to *WLO*. In theory, this could lead to a level of under-reporting, where The WildList may not accurately reflect reality.

The WildList lists those viruses that are actively spreading ItW. In researcher circles, to spread means to propagate. The words 'spread' and 'propagate' are used interchangeably. The operative word, however, is 'spreading'.

That alone should be all that is needed to refute this so-called confusion. Again, it is a matter of understanding what The WildList is supposed to represent. The WildList is not a list of *all* viruses found In the Wild. It is a list of those viruses that are found to be *spreading* In the Wild. It stands to reason that if a virus is detected and repaired without a problem, it is not spreading.

One argument against my last statement might be 'Just because one scanner detects and repairs the virus without a problem doesn't mean another scanner will detect and repair it without encountering any problems. Therefore the virus may in fact be spreading'. This is not only a valid argument, it is one reason why *WLO* has many Participants representing many different products. Given the geographical areas the many anti-virus products covers, somebody will surely report a virus that is spreading. The virus may escape the attention of one Participant, but is not likely to escape the attention of 54.

Of course, the way to eliminate any confusion is to redefine which viruses should appear on The WildList. Instead of listing only those viruses that are spreading, *WLO* could ask for reports of any virus seen In the Wild – even if it was seen only once.

This solution presents even more problems, however. The end result would be a very bloated WildList filled with inconsequential viruses (to most of the world, anyway). Once again, the WildList lists viruses that are spreading – those viruses the anti-virus industry agrees are of utmost importance to detect and repair.

Listing viruses that are immediately dealt with i.e. *not spreading* would only dilute the importance of focusing on (detecting and repairing) those viruses that are not readily detected and repaired. This type of solution is then unacceptable to those who find The WildList a valuable tool in its current definition.

'Only problem viruses are reported.'

Part II – In addition to the above Part I definition, a virus that was once considered to be problematic is no longer so because sufficient time has passed to allow most scanners to detect and repair it without problems. Thus, the virus is usually no longer reported. However, it is still actually out there. Even so, the virus may disappear from The WildList as early as one year after its initial placement. Once again, this could result in a WildList that may not accurately reflect reality.

A virus, once reported to *WLO*, remains on the list for one year. After one year, the 'identifier' of the Participant is automatically removed, unless the Participant specifically requests their identifier remain as continuing to report that virus. When only one identifier remains, the virus name falls from *the* WildList to the supplemental list. When no identifiers remain, the virus name is removed completely from The WildList.

The reality is that the default one year period is a sufficient amount of time for a virus to remain on The WildList. Furthermore, because the list is cumulative, the chances that a virus will remain on the list for longer than one year is highly likely.

Certification bodies generally test against a WildList that is no more than two months old. Due to the impact certifications have on the perceived viability of an anti-virus product, most anti-virus software vendors make efforts to maintain high levels of detection and repair of all viruses listed in *the* WildList.

In theory, this means viruses listed in *the* WildList will start to die off in great numbers just over two months after they appear on the WildList. After 10 months of a virus being such a high profile target, chances are the virus is not surviving with prevalence.

Nonetheless, *WLO* recognized this as a potential hole for newcomers to the anti-virus industry. So I am now going to play devil's advocate by presenting a solution. I do this not only because there may still be new anti-virus products introduced in the market, but also because I dislike the earlier 'In theory… ' statement.

'In theory' it would take only one infection to increase (locally) the prevalence of a virus. Based on the assumption that most existing scanners would detect and repair the virus without problem *by that time*, the repeated increase in prevalence may go mostly unreported. If this were the case, The WildList may *not* reflect reality.

In addition, the newer anti-virus products may not detect the very old WildList viruses. If such a product were to hold the majority of marketshare in a geographical region, a virus (detected and repaired by most other anti-virus products) may be allowed to exist with prevalence in that area of the world.

In order to fill this potential hole, the solution is to request reports from Participants who do not work specifically for anti-virus vendors. *WLO* has recently managed to accomplish this by implementing 'corporate reporting'. *WLO* suspects that corporate reporters will most likely report older viruses with more frequency than classic Participants are prone to do.

'The WildList lists only new viruses.'

Viruses that are new to each WildList are in fact viruses that are new to 'the scene'. The WildList rarely lists those viruses that are old and 'known' – e.g. viruses found only in privately-held libraries, or those already easily detected and repaired by most scanners.

Once again, a reiteration is needed of what The WildList is about. It is about listing those viruses and virus-like programs that are found spreading throughout diverse user populations. The WildList lists those viruses the anti-virus industry agree as important to detect and repair.

However, this explanation does not fully address why older viruses rarely appear on The WildList. As usual, an answer cannot be given in a single, simple, tidy little paragraph. In fact, there are several reasons why older viruses rarely appear on The WildList. The following are a few of the most common.

If a virus made it to The WildList at some point, lived its year, and then fell from The WildList (either to the supplemental list or completely), the chances of it reappearing on *the* WildList are greatly reduced. It does happen, however. There are several cases where an older virus makes its way back to *the* WildList. However, if it does not reassert itself, this does stand to reason and is a great compliment to the importance of The WildList.

> *"When a virus appears on The WildList, it becomes a high profile target."*

Within a short amount of time, most anti-virus products detect and repair the virus. Soon enough, the virus's death rate (the rate at which it is being detected and repaired) will outnumber its birth rate (the rate at which it is able to replicate itself further). For any virus, this is the ratio of disaster. As its numbers dwindle, it becomes less likely for the virus to resurge with prevalence (especially since the virus is already a very high profile target).

When the virus decreases sufficiently in prevalence, fewer reports are made to *WLO*. Eventually, the virus falls from *the* WildList. At that time, products are no longer required to detect and repair the virus in order to do well in a review or pass a certification. This does not mean, however, that anti-virus products then reduce their detection and repair capability. The detection and repair mechanism for the virus continues to reside in each product, and each product continues to detect and repair the virus (by this time) without problems.

To the user, an incident involving this virus may be completely invisible. Due to its seamless detection and repair, that virus' prevalence continues to be kept at bay. This ultimately reduces the possibility that the virus will once again re-emerge in sustained prevalence and reappear on *the* WildList.

If a well-known virus escapes into the wild and is immediately detected and repaired by 97% of the scanners, the virus does not fit the definition of In the Wild. While there is no doubt the virus was found In the Field, its existence would most probably be cut far short of it actually spreading. If somehow the virus did start to spread In the Wild, one or more *WLO* Participants would surely report it.

If a virus never makes it to The WildList to begin with, the chances that it will in the future are greatly reduced. There are several reasons why, the most obvious of which follow:

- the virus was never released into the user population and exists only in a closely held library or

- it is not a virus at all, rather it is a failed attempt at a virus or
- it has an inherent weakness; it simply would not survive in the field long enough to spread (thus becoming In the Wild.)

There are no solutions to this problem, nor are any necessary. The WildList works as designed. If an older virus increases sufficiently in prevalence, a *WLO* Participant will surely report it.

'The WildList is a list of viruses having the greatest reported frequency.'

A common misconception is that The WildList lists viruses which appear most frequently. This is far from what The WildList represents. Believing differently results in further misuse of The WildList.

Clearly stated within each WildList is the statement 'The list should not be considered a list of "the most common viruses", however, since no specific provision is made for a commonness factor'.

Common sense tells us that if something occurs with regularity it is a common event. However, in the world of anti-virus, there is little agreement as to what makes a virus the *most* common. Indeed if you asked the Participant who represents Japan which virus is the most common, his answer may differ from the person who represents Syria. Furthermore, it might depend on *when* you ask the question.

Also clearly stated in The WildList is that it is a cumulative list. Reports of a virus by a Participant are carried forward 12 months. As more reporters report the same virus, more identifiers are added. After time, it may appear that the virus with the most identifiers is the most common. This is not necessarily the case. More identifiers do not necessarily indicate a virus is the most common. A greater number of identifiers simply indicate the virus has, over time, been seen by more Participants.

To determine which virus is the most common, one must compile the individual frequencies per reported virus given by the individual Participants. Individual frequencies are part of what Participants include in their reports, however this data remains unpublished.

Even if such a compilation existed, it would still not clearly indicate which virus is the most common. It would instead give an idea of which virus is the most common in each geographical area of the world. More specifically, it would allow one to follow the rise and fall in the prevalence of any given virus as it makes its way from country to country.

As an aside, during November 1999, the two viruses with the highest reported frequency were W32/Ska.A (aka Happy99) and W32/PrettyPark.A. Of these two viruses, you may notice the former is indeed listed as the most frequently reported virus (see The WildList: Most Frequently Reported section, but also carefully note the description of *how* viruses come to be listed in the Most Frequently Reported section). The latter virus, while it appears in the Most Frequently Reported section, shows up towards the bottom of the list. Still, W32/PrettyPark.A had the highest reported individual frequency.

The point is that a virus listed on the WildList Frequency section is not necessarily the most common (i.e. that which is reported with a high frequency). Nor is a virus that is *not* listed in WildList Frequency section the least common.

'Names listed on The WildList do not match the names used by anti-virus products'.

When a virus appears on The WildList, it may appear under a name that only one anti-virus vendor uses. The name may not be recognized or used by any other vendor. The result is mass confusion! How are the anti-virus vendors who are *not* reporting the virus able to determine exactly *which* virus is being reported?

When a virus is reported to *WLO*, a participant will usually report it under the name displayed by their product. During collation of The WildList, the name and submitted sample are cross-referenced. If a different name exists – one that is recognized by the majority of anti-virus vendors – the recognized name is used on The WildList. The name originally reported may then become an alias.

The Participant is notified if the sample they have submitted does not match the name reported. They are made aware of the officially recognized name, and that the report is listed on The WildList under that official name. At this point, it is possible for the Participant to request that their vendor change the name in their product. By following through with the name change, the vendor has the opportunity to align their virus names with those used by the majority of the industry.

The naming problem more often rears its ugly head when an unrecognized virus name is reported to *WLO*. Since *WLO* has no way to prove (nor would it want to) that the name given by the Participant is incorrect, it must then consider the name given by the Participant *as correct*.

WLO uses the given name because it knows The WildList is also a tool for the user. *WLO* knows that if a vendor reports a virus (where no other vendor has reported the same virus), then that vendor (and therefore, that vendor's customers) must be seeing that virus *by that name*. Understandably, this method causes problems for those researchers who insist on eventually reassigning the virus with their own name.

In all cases *WLO* does its best to align the names listed on The WildList to those names that are officially recognized by the majority of the industry. Where this is not possible, however, the name with the greatest amount of exposure is used. It is a methodology used to benefit users. Unfortunately, conflicts arise when the virus is officially recognized and assigned a name that is very different from what is listed on The WildList.

Rather than The WildList changing what it lists from month to month to match names established later, including arcane names known only within anti-virus research circles, it would be better for researchers to consider the name used in The WildList.

It is important for vendors to accept the fact that The WildList is a tool initially conceived to benefit the user. As such, the reason why certain names are used (and will continue to be used) in The WildList is because they are the names that immediately have the most user exposure. For more information on the naming issue and how *WLO* manages things from an anti-virus research organization perspective, see http://www.wildlist.org/Naming_Policy/.

Conclusion

If it were not for reviewers, many of us would not have a clue as to the viability of a product. If it were not for reviewers, many products could possibly go unnoticed. If it were not for reviewers, a large percentage of the population might remain uninformed about how many different products are available. The WildList is a welcome tool for anti-virus software reviewers and certification bodies.

If it were not for users, reviews and certifications simply would not matter. The WildList is an excellent tool for the user. It gives users an accurate picture of which viruses are being found with prevalence. In combination with ItW certifications, it allows them to know how their products do against a *proven* threat.

The WildList is created by those who have a direct interest in keeping the computer virus threat under control (the Participant), and by those who rely on anti-virus software (the corporate reporter) as their first line of defence. If it were not for the Participants and corporate reporters, The WildList simply would not exist. However, because The WildList exists via so many different sources of input, it is an excellent and impartial tool valuable to all.

WLO realizes the positive impact it has in the anti-virus industry. The anti-virus industry also seems to realize and support the many good reasons for the popularity of The WildList, for the generally agreed-upon way The WildList is formed, and for the continuation and further evolution of the *WLO* project.

OPINION

Learning from Experience

Péter Ször
SARC, USA

The other day Eugene Kaspersky told me that a Russian magazine calculated how much money a single virus has the potential to generate for AV vendors around the globe. It is funny to think that an average macro virus could mean much more than our salaries for several months. Seriously, the anti-virus business is nothing like it was 10 years ago when I started to be interested in computer viruses. Or is it?

The Good Old Days

In 1990 I felt I was competing against John McAfee's *Scan* with my own program. *Scan* could catch about 40 viruses at the time and my product could detect and disinfect most of the same ones. A decade has passed and it turns out that I am still a competitor of the legendary software, regardless of the fact that not too many people know what John McAfee is doing nowadays and neither do they remember my first anti-virus effort.

I was lucky to have met Dr Alan Solomon during his active years in anti-virus development. My fear is that this tendency will continue and other heavyweight anti-virus folk will leave computer virus research during the next couple of years. I also think a few smaller vendors will merge into one big company realizing the needs of a bigger market and stronger resources.

It is always easier to predict the future if we take a look at the past. In 1990 there were so few viruses. The number soon exceeded the magical 300 mark and a little later there were thousands of them! A few names pop into my mind: Brain, Jerusalem, Stoned, Yankee_Doodle, Form, Tequila, Michelangelo, DiskKiller and DIR_II. Many of you will remember Ripper and One_Half and lots of others. It was so difficult to deal with the first polymorphic viruses. I remember looking at the MtE-generated decryptors for a week, it was really crazy! I laughed when one of my friends told me that computer viruses would give me a job for life.

I believe the virus of the decade was introduced in 1995. It was WM/Concept.A, the first in-the-wild macro virus to give many companies a new challenge. Before Concept several products had tried to follow the 'detection-without-repair' concept. They soon learned that disinfection is important, especially for corporations.

The Mighty Macro

Macro viruses appeared because the major operating system used on computers slowly changed from DOS to *Windows 95*. Several DOS and boot viruses were compatible

with *Windows 95*, but soone or later many of them became incompatible with it. However, and most importantly, the system reached a lot of homes on clone PCs and this meant that virus writers had all the resources to create new viruses on it. Developing a binary virus is much more difficult than writing a macro virus in *Word* Basic, VBA or VBS.

Win95/CIH caused major problems for many big corporations and home users. CIH was the first *Windows 95* virus successful enough to cause world-wide infections. CIH made history by being the first virus to wipe out the flash-BIOS code, making the hardware unusuable.

The author of WM/CAP.A (one of the most successful macro viruses ever created) soon got bored with macro viruses and created the first Win32 virus, Cabanas, at the end of 1997. New developments with Win32 ended up with the Win32/Ska.A (Happy99) Worm. Happy99 is virtually everywhere now, one year on from its creation.

The same idea was quickly introduced in a macro virus and many companies learnt about Melissa the hard way. Its author has created several in-the-wild 32-bit *Windows* viruses, but the problems he caused with Melissa were not comparable. There is a close relationship between macro viruses and 32-bit *Windows* viruses – the main environment of both is *Windows*. As *Windows* gained momentum in the marketplace more viruses have been introduced into it.

It is a long time since I heard about a new in-the-wild DOS virus. Macro viruses and Trojans are causing the problems for most of our customers these days. More and more corporations have to deal with 32-bit *Windows* viruses, too. At the time of writing the number of known 32-bit *Windows* viruses has itself passed the 300 mark. Does that mean that we will have thousands of them out there soon? Absolutely! There will be many new multi-partite viruses with VBA and VBS as well as 32-bit *Windows* virus infection capabilities. An anti-virus company has even been hit by a VBS creation and that shows the problem is already big enough. When *Windows 2000* reaches the market VBS viruses will assume real significance.

The Theory of Evolution

I think virus development is set to continue on the Worm front for the foreseeable future. Undoubtedly, virus writers are looking for methods to write real Worms that introduce themselves to remote systems without user intervention. The author of the BubbleBoy Worm has already found a successful way to do this. I believe the binary virus creators also will try to adapt this idea in order to create more successful Worms.

Recently, it was revealed that *Microsoft* has not won the 'monopoly' game, after all. There is a definite chance that other operating systems will profit from the impact of this. Personally I have never made any money out of my Unix knowledge (which is very little) and I would not be afraid to see *Windows* disappear from the market.

However, systems such as *Linux* are likely to be employed by more and more corporations. I am not talking about home users. I do not expect them to learn zillions of command-line parameters and walk around with a 1,500-page book entitled 'A Short Course on Linux'! In any case, this particular system will be used increasingly and this means one sure thing for anti-virus people. It is time to consider the *Linux* platform seriously and devote more resources to researching it. You can be sure that more viruses will appear on *Linux* if (and this is a big if) it reaches more homes. Since the executable COFF format is so similar to the Portable Executable format used by 32-bit *Windows* systems, it is reasonable to believe that cross-platform viruses will be developed for *Linux* and Win32 systems (including *Windows* CE).

Microsoft is dropping the support of Alpha environments with *Windows 2000*. That leaves *Intel* processors the major environment for all operating systems currently significant in homes and corporate environments. Win64 is knocking on the door and so are Win64 viruses. Imagine several AV companies still supporting low-end machines. It is funny to deal with the 64 KB limits when virtual memory managers are built into the operating systems. It is going to be no fun decrypting 64-bit viruses (which use 64-bit encryption keys by default) even on those 64-bit machines. A few Win32 viruses with MMX support already indicate the problem.

The evolution of computer viruses is certainly speeding up. Computation power is getting bigger and bigger and this leads to extremely complex polymorphic and metamorphic viruses (viruses that are based not on mutated decryptors, but on modularity).

Caveat User

Bearing all this in mind, what can the 'good guys', the anti-virus people, do in the foreseeable future? We need to work more on automation and spend more time developing generic solutions. This will give us enough time to handle the most difficult viruses most of the time. Virus labs will have to build test environments for testing Worms. One short year ago we could build a test system each time a new Worm appeared, but it is absolutely unacceptable now.

If you are responsible for keeping your corporation's computing environment virus-free, your goal should be to gain extensive knowledge on the security features of your operating systems and applications. There are still large corporations out there with many security holes in their systems leading to virus infections which should not happen in the first place.

It is imperative to keep updating your system's virus detection databases. Heuristics catch a lot of new creations, but you must use the very latest AV databases to get efficient protection. Many companies have 'crossed their fingers' hoping nothing bad happens, only to see thousands of their PCs infected with a virus that could have been caught just by using the latest update for their scanner!

A DAY IN THE LIFE

Viruses: A Way of Life

Eddy Willems
Data Alert International, Belgium

I remember the time when there were only 20 to 30 viruses around. I have seen old, familiar faces disappear in the last few years and new ones come along. I'm not a new one. I'm an old face who has spent a long time in the background. Some of you know me but for those who don't…

I was born in 1962. I am married to Nadine, a police officer – you could say we are both in 'security'. We have a son, Frank, who the fact loves that I'm working with computers probably because of all the games I get! I studied Computer Sciences at IHB and VUB (University of Brussels) and was employed as a Systems Analyst in 1984. In 1987 I worked at *Vaderlandsche* (an insurance company in the internationally known *ING* group) as a System Support Engineer and Security Officer.

In those days, I programmed in languages like COBOL, assembler and C. I also did some data recovery work. I became interested in viruses at the end of 1989. It all started with a diskette which was given to me by my former employer who said 'Try and test it, it has proved quite a challenge for our company doctor.' I didn't know that this would change my life!

It Started with a Disk…

Between 8 and 12 December of that year, twenty thousand envelopes containing a 5.25-inch floppy disk were mailed to computer users all over the world. The disk was labelled 'AIDS Information diskette' and encouraged the recipient to install it on a computer. Enclosed was a leaflet with a licence agreement urging the user to send US$378 to a post office box in Panama. There was a threat in the agreement that unspecified action would be taken if the appropriate fee was not paid. I received just such a disk.

The floppy itself contained a nice questionnaire which assessed the user's exposure to the real aids virus. Once installed, the program printed an invoice giving the address in Panama to which payment should be sent. At that time the installation procedure made modifications to the AUTOEXEC.BAT with the result that every time it was executed, a counter in another file was incremented.

After about 90 (random) counts the trigger activated. The file names in the root directory of the hard disk were encrypted and marked hidden. I made a program and a procedure to reverse that process at that time. It seems that I was the first in Belgium to have a solution for the AIDS Information diskette incident. It made me quite popular on TV and in some journals.

The writer of this Trojan had obtained mailing labels from the *PC World* circulation department. The case was solved by chance, when Dr Joseph Popp was stopped by a security guard at Schiphol Airport. He was extradited to the UK to await trial but his strange behaviour (wearing hair curlers in his beard etc) caused him to be declared unfit to be tried and he was returned to the US. He has since been found guilty of 'attempted extortion' by a court in Rome.

From that point on I began to gather information about computer viruses and anti-virus software. In 1990, by sheer coincidence, I made a modem connection straight to a US Unix-operated BBS. I was surprised when I recognized the operator's name: Sarah Gordon. In 1991 I became a member of *EICAR* and attended the *EICAR* conference along with lots of old friends. It was a strange feeling for me drinking beer (Belgium is renowned not only for its famous chocolates but also for over 500 beers, not forgetting Belgian fries… historically not a French invention!) with all those familiar names like Alan Solomon, Vesselin Bontchev, Frans Veldman, Paul Ducklin, etc…

Over the years I have maintained a reference library of software, books and almost everything that has been published in the anti-virus field. I have also been a reporter for the well-known WildList since 1995. That same year I started writing anti-virus-related articles for Belgian magazines. I also put up my Web site specifically designed to be the index to all anti-virus related pages and Web sites (http://www.wavci.com).

In 1996 I started conducting seminars and workshops about computer viruses. At that time *Microsoft* came to me and asked to write the 'Virus Article' for the *Microsoft Encarta Encyclopedia*. Since the end of 1996 I have been working as a Technology Consultant for the largest *Benelux* distributor of the former *Dr Solomon's Software*, now *NAI TVD: Data Alert International*. In this role I am responsible, together with eight other colleagues, for anti-virus consultancy work, support, training and research. *Data Alert* specializes in all security-related products. Finally, at the start of 1999, I took on the extra job of news editor for the *EICAR* magazine.

Pick an Average Morning

My alarm clock rings at 6am on a day in November 1999. I expect this day is going to be like any other as I get up and hop in the shower, before running to one of my many computers. Already, my email box is filled up with over 100 new messages and that's without looking at my second (*Data Alert*) email address (which I last checked at midnight). Trying to eat and read and answer the messages at the same time is not easy. I check my agenda and see two jobs for today. Two, hmm… there is definitely a third.

I jump into my car at half past seven and drive to a see a customer – a large, French-speaking bank. We have three official languages here in Belgium: Dutch, French and German. My native language is Dutch. It is what I call a site visit day. Sometimes I stay in the office to look into virus samples and problems from customers, sometimes I get out and do some training or some Virus Workshops.

My training courses are what I call really deep product training while the Virus Workshops are completely product-independent. The Workshop gives our customers (ranging from worldwide corporates to small businesses) an insight into the real threats which are around the dark corner.

I arrive at the bank in Luxembourg without a problem. It is a Virus Workshop this morning. A lot of questions are asked about the future. I explain that we are seeing a number of new problems this year and start talking about an increase of Worms, email propagation and VBS viruses. Then I go on to reiterate the importance of updating anti-virus products.

… and an Average Afternoon

My mobile phone rings at lunchtime. Our support department informs me of a problem that one of our customers has with a potentially new virus. They ask me to call in on my way back. So, at 2 o'clock I arrive at – let us call this 'Company 2', a Dutch-speaking one this time. The IT manager tells me that they've got some email problems whereupon I open my notebook, copy some samples down and started to analyse the documents. I'm assuming at this point that it is another W97M/Melissa variant, but a few minutes later I witness some interesting things.

This particular virus hooks the system event that opens documents in *Word 97* with the 'Document_Open' subroutine, thereby running its code. Another system event is also hooked – the closing of documents due to the subroutine 'Document_close' in the global template after infection.

There is also a self-check to verify if the local system has already been infected. It's a check for existence of a certain registry key. If this key is not found, the virus code uses VBA instructions to create an *MS Outlook* email message with the subject line 'Message from' (Username) and in the body 'This document is very important and you've got to read this!!!'. The first 50 listings from all available address books are selected as the recipient.

Furthermore, that is not the only payload. If it is 25 December this virus overwrites the existing AUTOEXEC.BAT with instructions to format the hard disk.

I also see a message box. After clicking OK on the dialog box, random coloured objects fill the document as an overlay. This reminds me of W97M/Pri. I call the Virus Lab in Aylesbury, UK. It transpires that I am the first in Belgium to run up against this one. They get an extra driver to repair it and the virus is given the name W97M/Prilissa.

Again my mobile rings. It is Rainer Fahs, Chairman of the Board of *EICAR*, wanting to discuss some small points concerning the upcoming *EICAR* newsletter and the next conference, to be held in Brussels in 2000. I arrive at the meeting place with Rainer at 6.30pm. We discuss some confidential *EICAR* stuff and I start for home to have dinner with Nadine and Frank. The next two hours are dedicated to my family. I can't resist it when my son asks if I will put him to bed!

After a while I hear the sound from one my computers upstairs: 'Incoming mail'… I look into my mailbox from the downstairs PC and see about 200 new emails. I start reading and answering them. I also look at the virus samples I received today. After replicating and classifying them I start to make them ready for inclusion on the most recent WildList.

I also take a look at two new magazines which came out today. One of them appears to be the December issue of a computer publication which features an interview with me! I rush upstairs to show Nadine. She's already asleep though, probably tired after chasing after some criminals today… honestly, it's an interesting case but that's a different story. I put the magazine away for tomorrow.

Midnight. Time to sleep. After finishing a very good book concerning hacking techniques I turn to my bed. My mobile rings at 1am. Oh yes, I forgot, *Data Alert's* 24-hour emergency line was forwarded to me today. It is a desperate customer who's seen some strange behaviour on his machine. After helping him with some advice it looks to me that he is infected with VBS/BubbleBoy.

Afterwards I try to catch up on my sleep again but I am still wondering about that third thing that I had to do today. I still can't remember it. At 2.00am I suddenly wake up… the third job: I was meant to be preparing a paper!

FEATURE SERIES 1

Malware Do You Want To Go Today? Part 1

Eric Chien
Symantec, Europe

As the *Virus Bulletin* conference in Vancouver wound down and the last beers were poured, a question still remained to be answered. At the final Speakers' Panel, Steve White of *IBM* asked, 'In the next year, do you think the virus problem will be worse, better, or the same as this year?'. Glancing around the room of three hundred people, hands shot up in response. What did the overwhelming majority in attendance vote? Worse. The virus problem will be worse next year than it was this year. I agree, and I think most other researchers agree too.

However, differences of opinion arose and accusations of optimism (dare we be optimistic?) were declared when I suggested 'Yes, the virus problem will get worse, but anti-virus products and awareness is getting better and thus, it will even out, and things will be the same'. So, to back up my statement, let's predict what the major threats of the next year will be and also what anti-virus vendors are doing today to protect us from the threats of the future.

Network-aware Malware

We continue to talk about network-aware malware (malicious software that utilizes a network) and how we are at a critical juncture in virus history. Well, that is no longer true. We have passed the critical juncture with W97M/Melissa variants, the ExploreZip Worm, password-stealing Trojans, Back Orifice, VBS/Freelinks, the PrettyPark Worm, the Happy99 Worm, VBS/BubbleBoy, W32/FunLove.4099.

The list goes on and on. Adding a network component to a piece of malware has become almost standard and will certainly become so in the next year. By the end of next year, I believe the majority of the top ten infectors on the WildList will be network-aware malware.

Why? The answer is simple. Samples of malware that use the Internet and local area networks spread significantly faster than previous infectors that were not network-aware. Furthermore, they have no geographic boundaries and require no human intervention.

Even the most common Boot viruses of yesteryear required that individuals exchange floppy diskettes. Ask yourself this; how many times today have you passed a floppy diskette to your colleagues as opposed to sending them the files via email? Almost never, I bet. When was the last time you sent a DOS executable file to a friend? Classic infectors depend on just this type of human intervention to spread from one computer to another, and in doing so they are limiting themselves to the boundaries both of time and of geography.

However, network-aware malware that can send itself using your email is no longer limited by geographic boundaries and no longer needs a human to send an infected file to a friend. The malware does all that for you! Thus, the malware spreads worldwide within hours as seen with W97M.Melissa. Other examples include the PrettyPark Worm, which sends sensitive information to an IRC channel, and remote access Trojans that can turn on your microphone and webcam.

The days of worrying about a payload that simply deletes a file on your computer are over. We must worry about payloads like that of the ExploreZip Worm which can affect an entire network of PCs rather than just the infected machine. We must worry about payloads that ship off confidential files to some anonymous email address or public Internet site. In the next year, we must expect malware writers to use even more network features than we have seen today.

Windows Viruses

DOS file viruses, although still discovered today, are rarely a threat. Boot viruses verge on extinction. However, *Windows* viruses are turning into the new macro virus with regards to prevalence. Granted while we do not yet have non-preprogrammed mutating and mating *Windows* viruses, in the past we only discovered one or two new *Windows* viruses a month and now today we see one or two new ones a day. The number of *Windows* viruses is definitely rising.

Why? Again, it is a simple answer. Viruses spread on the platforms and mediums we use. If we all used Java applications, there would be many more Java viruses. If we all used Macintoshes, there would be more in-the-wild native Macintosh file viruses. However, corporations and home users overwhelmingly use *Windows* operating systems. Thus, *Windows* viruses are becoming extremely prevalent.

What makes it worse is that *Windows* viruses are very difficult to analyse. Even with the best tools, analysis can take hours. WinNT/RemoteExplorer took days. Imagine a piece of network-aware malware, which spreads in hours, taking days to analyse. There are *Windows* viruses that utilize anti-debugging tricks, MMX instruction sets, and obfuscated entry points. Moreover, almost all take advantage of undocumented *Windows* features.

These techniques all make *Windows* virus analysis difficult. *Windows* Worms are often coded in a higher level language, make use of standard libraries, and thus, are huge in size. Inspecting every byte of code is tedious and difficult. In

addition, *Windows* viruses can go memory resident and no current anti-virus product has a generic *Windows* memory scanning engine. As things stand, if you are infected with a memory resident *Windows* virus and you scan your system, you have just given that same virus the potential to infect every file scanned.

Scripting

Virus writers try to write viruses in every possible language. We have seen VBScript viruses, JScript viruses, and IRC script Worms amongst others. Script languages are generally much easier to program in. Virus writers without much knowledge can, for example, create a batch file virus in a few hours.

Creating viruses using *Windows Scripting Host* and other scripting languages is only a bit more difficult. There is generally no need either to understand the inner workings of the *Windows* operating system or for undocumented features. In addition, operating systems are beginning to support more and more of these scripting languages.

Almost a year ago, receiving a VBScript file was unheard of and figuring out how to run it would have been almost impossible. Even today, people generally do not transfer VBS files from one to another, but that does not matter, because new *Windows* operating systems will recognise and run VBS files for you with a simple double-click. And we know, the temptation of double-clicking on a file that offers 'Free XXX links' is often irresistible.

Programmability

Almost every application program of tomorrow will be scriptable or programmable, and thus will become susceptible to viruses and their payloads. IRC clients support a script language, IIRC. This script language allows you to create scripts to automate simple tasks such as notifying you when your friend gets on-line as well as to create complex programs such as a whole AI robot. Unfortunately, somewhere in between the two is the ability to create IRC script Worms. IRC script worms would not exist without the programmability of IRC clients.

Microsoft Outlook can be invoked to send out an email with only a very few lines of code (so few you can count them with your fingers). As we are now aware, *Microsoft Word*, *Excel*, *PowerPoint*, and *Access* all have VBA support allowing for macro viruses. What is worse, VBA is not limited to *Microsoft* products. *AutoCAD* applications have VBA support and so does the latest version of *WordPerfect*. *Microsoft Outlook* already has VBScript support with VBA to be added in the near future. In addition, *Lotus Notes* supports *Lotus* scripting.

While this programmability does allow us to automate tasks and create ever more robust and interactive programs, it also allows malware to utilize the same features. We must count every application that has programmability as a potential target for malware. The unfortunate truth is this – if you can program an application to do a task, so can a piece of malware.

Malware and Security Exploits

While the ability to create malware that would activate simply by reading an email has existed for some time, no proof of concepts were discovered until VBS/BubbleBoy (see *VB*, December 1999, p.6). VBS/BubbleBoy used a *Microsoft* exploit that allowed the creation of files on a system when simply reading email (no need to detach and run an attachment).

However, the usage of security exploits in general is nothing new. Back in 1986 the Morris Internet Worm utilized four different security holes to spread itself to more than 10% of the Internet at that time. In recent times, the *Windows* viruses Bolzano (*VB*, September 1999, p.10) and FunLove (December 1999, p.3) patch two bytes in the *NT* kernel causing all ACLs to be disabled. This takes advantage of an exploit which allows the subsequent modification of NTOSKRNL.DLL and NTLDR without *NT* realizing the kernel has been modified.

We can only expect virus writers to jump on the bandwagon and attempt to utilize even more security exploits. This will allow malware to spread further, no longer restricted even by security boundaries.

Ubiquitous Infrastructure

Big words. Simple idea. Everyone is using the same type of hardware, the same programmable applications, the same executable file formats, the same networking protocols, and by next year maybe even the same mousepad. All right, maybe not the same mousepad, but you get the idea.

With everyone utilizing the same computing system, if a piece of malware can infect one system, it can easily and equally efficiently infect every other system out there, because they are all the same. Carey Nachenberg's keynote speech at VB'99 outlined this fact and included a demonstration that showed us how ubiquitous infrastructure aids malware in its spread. Instead of diverging, these days computer systems seem to be converging into a single genetic type. If you figure out how to kill one, you figure out how to kill them all.

Summary

The number and types of threats seem daunting. However, that is nothing new. This industry has always been in an arms race – anti-virus vendors versus virus writers. In the end though, I believe that the anti-virus vendors are actually ahead in the race. Unfortunately, you will need to wait to find out why. In the February issue of *Virus Bulletin* I describe the range of solutions that are either available now or currently in development. From automation to education, there are ways to combat these pending threats.

FEATURE SERIES 2

Lotus Notes and Email Risks – Part 1

Martin Overton
ChekWARE, UK

The re-emergence of Worms earlier this year, coupled with the current tendency of virus writers to make their progeny 'Internet-aware' and with the growing trend of companies rationalising their software systems (known as standardising) means that the risk from viruses targeted at a particular infrastructure has increased dramatically. Many big corporations and smaller companies are standardising on *Lotus Notes* as their Groupware solution. It is estimated that there are 35,000,000 licensed *Notes* users worldwide (October 1999). Let's have a look at the strengths and weaknesses of this approach, specifically the email features.

Let me make one thing very, very clear. *Lotus Notes* has always had security built in and is, to the best of my knowledge, the most secure Groupware solution currently on the market. This does not mean, however, that it cannot be attacked; it just means that if it is properly installed and configured that the risks of attack are minimized, or in some cases neutralised.

One thing you must be aware of though is that, by default, *Notes* client installations (straight from the box) have most of the excellent security facilities turned off! I wonder just how many companies are running *Notes* clients in this way?

Let's get down to business. I will start by covering the potential risks (not touching on hoax viruses which, although an email-related problem, are beyond the bounds of this article) of using *Notes* clients in its 'out-of-the-box' state i.e. worst case scenario, and then move onto how these risks can be addressed, removed or where no solution is available, at least understood.

I believe that at the moment email (both attachments and scripting) is the biggest threat to users, and this shows no sign of changing soon. Indeed, the recent appearance of 'BubbleBoy', 'MyPics' and the numerous new Melissa variants indicates that this vector has become the main choice for many virus writers and other malcontents.

What are the Risks?

Well, there are two types of threat here, existing viruses and *Notes*-specific threats.

The risk to *Notes* users from existing viruses (macro, executable, boot, etc), which makes up the largest slice of the *Notes* email risk pie, can generally be summed up in one word; attachments!

Let's look at this in more detail:

Files (EXE, COM, DLL, SYS, SCR, HLP, etc)

The main threats here appear to be (at the time of writing) Worms such as Win32/Ska and Win32/ExploreZip, and a myriad of Trojans such as the ubiquitous *AOL* password-stealers. As this article goes to press, the trend seems to be accelerating, Win95/Babylonia being the latest example. The original file infectors still seem prone to slow propagation, unless they get lucky, and this is borne out by the low figures found in the *Virus Bulletin* Prevalence Table.

Boot Sector Viruses (MBR and DBR)

These are still a problem, although the number of outbreaks is still small, and the risks in an email scenario are minimal. This class of virus can still be sent as an attachment, though this requires a metamorphosis into a 'dropper', 'DEBUG Script', disk image format or other intermediate state.

Macros (*MS Office*, *Word Perfect Office*, etc)

Recently, macro viruses have been seen to account for around 80% of reports in the *VB* Prevalence Table. They present the biggest threat to companies, since *Office* files are passed around with, in too many cases, wild abandon.

This is compounded by the number of people who use *Word* as their default *Notes* email editor. Either that, or they blindly 'launch' attachments in their native *Office* applications. If the viral macros are not intercepted by any AV software, and they are run, the *Office* application environment becomes infected.

Scripts (JavaScript, VBScript, *WSH* files, etc)

VBS/BubbleBoy has shown that the scenario I outlined in my VB'99 paper is valid, although that Worm is targeted at *Outlook*/*Outlook Express* rather than *Notes*. *Notes* can also run code when an email is either pre-viewed or opened and therefore could be used to launch a similar Worm without the use of any attachment.

HTML sent to *Notes* could have VBScript (including *WSH*) or JavaScript embedded in it, which will trigger if opened in a browser that supports it. *Notes* allows you to choose which internal browser to use with *Notes*. *Lotus*/*IBM* offer their own browser, or you can choose to use either *Microsoft Internet Explorer 4.x* (or later) or *Netscape 4.x* instead.

It is certainly possible that a virus such as VBS/BubbleBoy could trigger when an HTML attachment is launched from *Notes* into *Internet Explorer* (always assuming that the remaining requirements are also met for it to trigger either partially or completely).

Extract from 'Viruses and Lotus Notes: Have Virus Writers Finally Met Their Match?'

'*Because of the power of* Notes *macros and other scripting languages, which are either part of* Notes *or are supported by* Notes *this auto-launching can occur!*

What's worse is that infected files using auto-launching can also be despatched to hundreds of users via email before they are detected, at the least this can cause a mail storm, at worst it could cause loss or theft of critical data and even publishing of sensitive data to the internet as in the PolyPoster and Caligula viruses.'

As for the *Notes*-specific threats – let me make it quite clear that, as at the time of writing there are no *Notes*-specific viruses known. Current threats are limited to Trojans, possible Denial of Service attacks and mail bombs. I do believe that viruses and Worms can be created within *Lotus Notes*. Let's look at the risks of each function/threats:

Trojans

In earlier versions of *Notes* (pre-*4.5*) Trojans, unlike mail bombs, required assistance from the user to trigger. This does not make Trojans any less dangerous, or less likely to be activated, as the number of victims of Trojans and Worms can attest to. With *v4.5* or later, this user assistance is now no longer required; simply opening (reading) or previewing a *Notes* email (sent from another *Notes* client) can launch an attachment or run code.

The features employed to create Trojans are everyday *Notes* elements which users handle without a second thought. The main *Notes* features open to such abuse are Buttons and Hotspots (popups and action hotspots). Since the arrival of *Notes v4.5*, the latter are a particular danger because they also allow the use of @Commands. Before that version, popups could only be used to display help text – now you can attach more than just *Notes* formulae to Action Hotspots; this can include LotusScript or JavaScript.

Mail Bombs

There are currently two main types of mail bomb: stored forms and self-launching OLE objects. The former will *force* a document into a stored form by saving (and sending) that stored form with the document. This is particularly effective when used with a 'computed for display' type field in the stored form. This can be used to initiate a predefined string of events, benign or devastating depending on the author's intent. Many mail bombs will go off with (little or) no user intervention. Merely opening one up is sufficient – as simple as viewing or previewing the email.

LotusScript

This is, in many ways, very similar to *Microsoft's* VBA. This, I believe, will soon give rise to LotusScript viruses, Trojans and Worms. LotusScript can include calls to external files, *Notes* APIs and *Notes* functions. This can readily be used to manipulate the user's mail file and any other databases that the user has manager rights to. By default, a user has manager rights to their mail database. It is possible that LotusScript could become the Achilles' heel of *Notes*, as VBA is to *Microsoft Office* applications, and we are all only too aware what that brought forth! It is rumoured that *Lotus* will be ditching LotusScript in the next release of *Notes* and just supporting JavaScript.

Indeed, I have already seen code samples that do run when an email is opened (read) or even previewed in the *Notes* client preview pane.

Stored Forms

Stored forms were first introduced in *Notes v2.0* and have since been considered a security threat by many *Notes* administrators. Stored forms can contain Formulae, JavaScript or LotusScript that can be triggered when the email is opened or even previewed.

OLE

According to *Lotus*, '*Object Linking and Embedding (OLE) is a technology that lets you share data between applications and is supported for Windows and Macintosh. OLE lets you link or embed data from other applications, such as a 1-2-3 chart, Word Pro document, or Freelance Graphics presentation, in a Notes document. You can embed or link part of a file or a whole file. You can also embed a new object in a Notes document and use the object's application to enter data in Notes. For example, if you have 1-2-3, you could create a blank 1-2-3 worksheet object and enter 1-2-3 worksheet data in a Notes document.*'

Formulas and Field Formulas

Lotus defines formulas thus – '*An expression that has program-like attributes; for example, you can assign values to variables and use a limited control logic. Formulas are best used for working within the object that the user is currently processing. The formula language interface to Notes and Domino is through calls to @functions.*

You can write formulas that return a value to a field, determine selection criteria for a view, create specific fields in a form, determine the documents a replica receives, help users fill out a document, increase database performance, and create buttons or hotspots.'

In Summary

There is certainly a possibility that a Melissa-type email Worm/virus can be written to target *Lotus Notes*. All the functionality is there and in its 'out-of-the-box' state *Notes* offers little resistance to these threats. The second part of this article, to be featured in next month's issue, will deal with what can be done to minimize or neutralise the risks outlined above.

TUTORIAL

The Final Frontier?

Iñaki Urzay
Panda Software, Spain

On 8 November 1999, virus writers took another step forward in their search for new techniques and infection methods. This was the day a new virus – VBS/BubbleBoy – was discovered. While it is not a technically complex virus, it has opened up a whole new means of infection by using scripts in HTML format, email messages and an ActiveX vulnerability in systems with *Windows Scripting Host*.

This article highlights the fact that it is no longer necessary to open an attached file to become infected by a virus. Simply opening a message or having the AutoPreview option enabled is all that is needed for a virus to attack. Some of us saw this coming a while ago but others said it would never happen, leading users to believe that infection was impossible unless an attached file was opened.

BubbleBoy incorporates an important feature that has not been emphasised in most articles published to date. As mentioned already, the virus is not located in an attachment, but rather in the 'message body'. Special attention should be paid to this term. It should be taken into account that, at present, the vast majority of email anti-virus products do not scan the message body. It is not enough to say that an anti-virus product scans inside the message body, as many messages may have not just one body, but two or three.

An *Exchange*/*Outlook* message contains fields with certain properties (subject, from, etc.). There is a property called PR_BODY, where the message body is found in plain text. In another property, PR_RTF_COMPRESSED, the same message is found in RTF format, and in *Outlook 98* and *Outlook 2000* a third property called PR_BODY_HTML has the text in HTML format.

It is not enough simply to scan the PR_BODY property in search of UUENCODE encoded files. It is also necessary to scan the message body in RTF format in case this type of file includes *Word* macros inserted either by the manufacturer or some kind of vulnerability or trick. Lastly, and most importantly, it is essential to scan the body in HTML format, as this is where many infections are expected to be produced in the future. Furthermore, applications that receive and save messages in their original MIME format, such as *Outlook Express*, incorporate several fields in both ASCII and HTML format, which should all be scanned.

VBS/BubbleBoy is the first virus to use this means of infection effectively, but it will not be the last. The moment email messages can be written in HTML and include scripts, they automatically incorporate all the problems normally associated with HTML pages, with the added disadvantage that they are sent to victims without their having solicited them previously. This, together with the fact that they can be automatically viewed by means of the AutoPreview option, is what makes them so dangerous.

Once a user receives a message infected with a script virus integrated in the message body, it is not possible to delete it without being infected. The action of clicking on the message with either mouse button produces a message and the corresponding infection. Therefore, the only way to remove the message is to delete it after having disabled the preview option. What is worse, you can also be infected simply by receiving the message or accessing a folder (if the virus-carrying message is there), as it will be viewed as soon as you enter the folder in question. To avoid infection and prevent scripts from executing malicious actions, ActiveX controls incorporate these safety mechanisms:

- Digital certificates: these are certificates issued by authorized companies that assure that the control cannot cause damage to your system.
- Controls marked as being safe when initialized from a script. The manufacturer of the control guarantees that its control will not cause damage to the system when initialized by a script.
- Controls marked as being safe when run from a script. The manufacturer of the control guarantees that no script will cause damage to the system or obtain information through its control.

In theory, a control that complies with these three requirements can be used by a script without problems. Bear in mind that the first of them is issued by an authorized company but the other two are guarantees provided by the maker of the control.

In principle, you may think that this system of providing guarantees poses no real problem, as we instinctively think of important manufacturers like *Microsoft*, *IBM*, etc. as major companies that we can trust in. However, the creation of ActiveX controls is not restricted to large organizations. Anybody can develop an ActiveX control and mark it as safe to be initialized or run from a script. To do this, a developer only has to create two entries in the Registry or implement the IObjectSafety interface. This way, you cannot even be sure that it was the manufacturer who marked the control as being safe. Any program or user with access to the Registry can convert an unsafe control into a safe one, constituting a security hole in the system.

Controls that are not downloaded from the Internet but placed on the system when installing new software from a CD (which means they do not require certification) are another problem. These controls are often created by small manufacturers or amateur programmers, who distribute

them as Shareware or Freeware. Even if you do not download software from the Internet, if an ActiveX control is integrated in a Web page your browser will not check it for certification before installing it. This means that it will stay there until it is uninstalled and may be used by a script integrated in an email message to gain access to your system. Even if a control is marked as safe, the manufacturer may not have actually dedicated the necessary time and resources to verifying that this is true.

Our systems contain controls created by the most prestigious of manufacturers. Although they are not used directly to create viruses, they are security holes that malicious users can use to obtain confidential information or carry out destructive actions. For this reason, creators of ActiveX controls should make sure their controls prevent access to the following before marking them safe for scripting:

- Creating files using a name provided by a script
- Reading files using a name provided by a script
- Inserting information in the Registry using a name provided by a script
- Obtaining information from the Registry using a name provided by a script
- Running API functions using information provided by a script
- Creating or manipulating external objects using identifiers provided by a script

Even if a control complies with these, it should limit access to the system as much as possible. A control *could* allow a script to write in a file in the temporary directory and still be considered safe. However, if another control allows modifications to be made to the directory, a script could use both controls to overwrite a key system file and get round the security measures. Users should never reduce the level of system security and allow scripts to run controls that are not marked as being safe. If this happens, any script integrated in a message, which could come from anywhere in the world, could take control of the entire system.

We have come across diverse controls designed by different manufacturers, widely employed by users the world over, that do not offer sufficient guarantees of security. The use of these controls could allow malicious users to do the following, among other things:

- Automatically launch *Internet Explorer* and load a Web page
- Open any *Windows* Help file, with the risk that this could be infected (W95/Babylonia)
- Access a serial port (and therefore a modem)
- Access the messaging API (MAPI)
- Access databases in SQL systems.
- Confuse users by displaying false dialog boxes or modifying parameters in the sound system

While manufacturers of controls should dedicate more resources to these security problems to avoid their controls being used by malicious scripts, the problem may be more complex. We should start by asking – what is the point of making script-proof controls when all that is needed to attack a novice user are two instructions written in Java Script or Visual Basic Script?

```
while(1)
    alert("A window that cannot be removed");
```

Perhaps we should consider whether it is useful to allow messages to include scripts when for security reasons these can do practically nothing. In other words, what good is it to integrate scripts in messages if you cannot access the file system, the registry or the *Windows* API? It is even inadvisable to display windows on-screen since, as seen above, they can be used to block novice users' systems.

On the other hand, scripts in HTML messages are not the only means that viruses can use to be executed without having to open an attachment. The *Microsoft Exchange*/*Outlook* applications use the *Windows Messaging* system to transport and store messages through the MAPI. This system is based on COM and structured storage, in which folders, messages, etc. are objects and where each of their characteristics is saved in a property.

In the case of messages, the PR_MESSAGE_CLASS property indicates the type of message in question, which is normally IPM.Note. However, if you create a message with an attached file and assign the PR_MESSAGE_CLASS property the value IPM.Document, the *Exchange* or *Outlook* mail reader will automatically run the attachment. Although this can be programmed, the easiest way to see it is to drag a document from the *Explorer* to an *Exchange*/*Outlook* folder. You will see that the mail reader will run it upon opening the message, without first opening the attached file. In fact, the message appears without the mark that indicates the presence of an attachment, which may lead the user into believing it is a simple text message and opening it without taking further precautions. This behaviour varies depending on the client version used.

Luckily, these messages cannot be sent with this property, which greatly reduces the risk of direct infection. However, messages of this type may be created in public folders, which, as you know, can be accessed by all networked users. In addition, if the infected user is an administrator, a message of these characteristics may be created in all the mailboxes of a particular server, which would ensure widespread infection.

In short, the only way to combat these new means of infection is to have permanent, effective protection systems that are specifically designed for use on the messaging software installed on your computers. These systems should be capable of both intercepting the message body in all its different forms (ASCII, RTF and HTML) in real time, and disinfecting it before the user has had the chance to open the message.

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/

US subscriptions only:

VB, 50 Sth Audubon Road, Wakefield, MA 01880, USA

Tel (781) 2139066, Fax (781) 2139067

END NOTES AND NEWS

A two-day course entitled Practical Anti-Virus will be run by *Sophos* on 25 and 26 January 2000 at the organization's training suite in Abingdon, Oxfordshire, UK. For further information, or to reserve your place, please contact Daniel Trotman at *Sophos*; Tel +44 1235 559933, fax +44 1235 559935, visit the company Web site http://www.sophos.com, or email courses@sophos.com.

The ninth annual *EICAR* conference, also known as the first European Anti-Malware Conference, takes place in Brussels, Belgium, from 4–7 March 2000. For further information, to place your booking reservation or to order a timetable of events visit the *EICAR* Web site at http://www.eicar.dk/.

Now in its fifth year, InfoSecurity Europe is the largest dedicated IT event in Europe. **InfoSec 2000 will take place at the National Hall, Olympia, London from 11–13 April 2000.** The show includes exhibitions and talks on various subjects including virus protection, firewalls, network security, e-commerce and Web security. There will also be a series of 46 free on-floor seminars on topics such as *Windows 2000* and *Linux*. For more details contact Yvonne Eskenzi; Tel +44 2084 498292 or email yvonne@eskenzi.demon.co.uk.

Network Associates Inc* announces the release of *Dr Solomon's Groupshield for Lotus Notes v4.5, the only groupware anti-virus software, the company claims, that fully supports *Lotus Notes R5*. The product is centrally controlled and includes NAI's ViruLogic technology. For more details contact the sales team at *NAI* in the UK; Tel +44 1753 827500.

The fourteenth annual *Vanguard Enterprise Security Expo 2000* will be held at the Atlanta Hilton and Towers, Atlanta, Georgia, USA on 15 and 16 May 2000. Details can be found at the Web site http://www.vipexpo.com or contact *Vanguard*; Tel +1 714 9 390377.

Network Security specialist *CenturyCom* announces the formation of its so-called 'Millennium SWAT Team'. It plans to provide 24-hour rapid response over the holiday period to problems directly or indirectly linked to the date rollover. *CenturyCom*, in association with other professional bodies like *Checkpoint* aims to plug any holes created by systems failure. For more information contact Clive McCafferty; Tel +44 1635 295500.

Romania is years behind other Western administrations in that none of the country's money is stored on government computers. This worked in the country's favour when, **at the beginning of November, hackers broke into the Romanian Ministry of Finance's Web site** and uploaded a personal copy of their own site. Laws and taxes were tampered with, including the creation of a tax on stupidity and the modification of the leu/dollar rate from 18000:1 to 1:2. Officially, no computer fraud was performed and, given the financial situation in Romania, some TV channels and national newspapers even hailed the hackers as friendly patriots.

It's catching! **A *Symantec* distributor in Croatia mass-mailed material infected with VBS/Freelinks to its customers recently.** This one could run and run… Also, the 'silent but deadly' virus is on the up, emphasising the importance of regular updates to anti-virus software. **W97M/Thus, alias the Thursday virus, shows no sign of infection until it activates on 13 December and deletes all files from the C: drive.** This macro virus spreads via *Word 97* and *Word 2000* files downloaded from email attachments or infected floppies or CDs. Most AV software updated since September 1999 will detect and repair it but always make backups!

While it is not the first and certainly will not be the last, *Computer Associates* is the latest on *Virus Bulletin*'s name and shame list. ***CA* has placed ads in *PCWeekAsia* for two consecutive months featuring misrepresentations of the Comparative Review test results and VB 100% awards.** Once again we must stress, the VB 100% award is platform-specific and dated for obvious reasons. *CA* omits any reference to platform or specific tests in its claims to be '100% effective'. Users be warned – detection capability and effectiveness are two very separate things.

In mid-December 1999 *Data Fellows Corporation* changed its company name to *F-Secure Corporation*, capitalising on the firm's internationally known *F-Secure* product line. *F-Secure Corporation* continues to house its headquarters in Espoo, Finland.

An updated, 10 year ***Virus Bulletin* back issues CD** encompassing all *VB* issues from July 1989 through to December 1999 is currently being finalised. Once pressed, the CDs will be sent out to all subscribers free of charge. Watch this space for more information.