ISSN 0956-9979 NOVEMBER 2002

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **A slap in the face:** Linux/Slapper is one of the most significant outbreaks on *Linux* systems to date. Frédéric Perriot and Péter Ször have all the details on p.7.

• **Gone fishin':** Pete Sergeant continues trawling through the hoaxes in his quest for automated hoax detection and details three tools of his own creation. See p.14.

• **Are you being served?** 21 products line up for a grilling – this month on *Windows Server 2000*. See p.16.

Virus Bulletin thanks the sponsors of VB2002:




CONTENTS

COMMENT

Talking Dialogs

“ Have we ever considered that the software designed to protect users was not created with the users in mind? ”

Let’s talk dialog … specifically, alert dialog boxes. Are the messages we give users in keeping with today’s threats or are we trying to make old messages fit new paradigms?

During the height of the Klez epidemic, its penchant for spoofing the From address resulted in an increase in the number of enquiries to corporate help desks. Even though the corporate systems weren’t infected, the confusion generated by the technique exasperated and sometimes overwhelmed the help desk staff. Thus, although the primary source of the impact may have been unprotected home users, it was often the corporates that bore the brunt of it. Clearly, we cannot simply ignore the impact on the home user if it’s going to affect our business, yet I often wonder if we aren’t devoting most of our talented resources to enterprise development.

Some of us grumble that end users are unaware of the risks, too stupid to protect their systems properly, or too willing to click on anything that crosses their path. We sometimes judge them for their panicked behaviour with the SULFNBK.EXE hoax and wonder why on earth they don’t take real virus alerts as seriously. Have we ever considered that the software designed to protect users was not created with the users in mind? Are we providing alerts that are too often confusing, misleading, or simply inappropriate? Worse, have we created a situation in which users are losing faith in their protection, causing them to abandon it all together? One popular anti-virus scanner (designed specifically for home users) begins installation with a prompt to uninstall *ZoneAlarm*. The same alert goes on to warn, ‘If you choose not to remove the incompatible software your system may not function properly.’ If users follow this advice, they lose a valuable piece of protection. Wouldn’t it be better to provide information on how to achieve compatibility? Or better yet, work to fix the problem? At the very least, admit that the *AV program* may not function properly, rather than implying that *ZoneAlarm* is responsible for system malfunction. *ZoneAlarm* can hardly be described as obscure and it should at least be *hoped* that a significant portion of home users are running it – and they should be encouraged to keep doing so.

One of the more confusing dialogs occurs when a worm or Trojan is detected. One vendor reports that it ‘cannot repair file’ and recommends that it be quarantined after which subsequent updates will try to clean and restore it. Another reports that the file cannot be cleaned and should be deleted, then restored from backup. Both of these alert dialogs leave many users with the impression that the virus has been detected in a necessary file, and this misconception causes unnecessary anxiety. Wouldn’t it be preferable, when detecting a worm or Trojan, simply to let the user know that the entire file is malicious, that there is nothing to clean, and thus cleaning consists of deleting?

An entirely different problem arises with Sircam. After the user diligently follows the advice of their anti-virus software and quarantines or deletes the worm, their system is often left incapable of launching executables because a simple registry edit was not included in the cleaning routine. Such impartial disinfection makes little sense from a development standpoint, and can cause major headaches – and expense – to inexperienced users who all too often turn to computer repair shops for help. The end result is usually a complete reformat of their drive and subsequent loss of data. While we might be tempted to shake our heads at the ‘ignorance’ of the tech or user responsible, how often do we reflect on how our own actions might be a significant contributing factor?

Finally, perhaps it’s time to tighten up on the use of the words ‘virus’ and ‘infected’ in alert dialogs. Judging from the many newsgroup posts on the subject, certain products’ detection of exploits and advertising ploys such as JS/NoClose may be causing more harm than good. Maybe it’s time to consider adding a new category of alert dialogs, specific to script bombs and other non-viral (albeit unscrupulous) advertising ploys. Certainly the tactics used by many pop-up enthusiasts can extend beyond merely annoying, and protecting against them might be a nice option, but to scare users into reformatting their drives because the ‘virus’ keeps coming back is another thing altogether.

Mary Landesman, Antivirus Guide, About.com

NEWS

Addendum: June 2002 *Windows XP* Comparative Review

In the June 2002 comparative review of anti-virus products for *Windows XP* (see *VB*, June 2002, p.19), we stated that W32/Nimda.A samples were missed by *F-Prot 3.12* 'due to extension issues In the Wild on access.' The files in question were the EML files dropped by Nimda. *VB*'s documented testing procedure involves the opening/closing of tested files and, for practical reasons, does not include the execution of any malicious code. In the vast majority of cases such methods are sufficient to trigger a reaction from tested products. However, it has been drawn to our attention that the on-access protection implemented in *F-Prot* purposely ignores the opening of an EML file as a non-threat event (treating such a file as a container) – yet, if an infected EML message is accessed in the real world (an attempt made to execute its contents), the product *will detect and block* the execution of the malicious code. We have tested the claim and are happy to report that, although the product did not detect Nimda's EML files, *F-Prot* users relying on the on-access protection against W32/Nimda.A are safe ▮

Moth-Eaten Software

A warning issued by Israeli security firm *GreyMagic Software* last month revealed a total of nine vulnerabilities in *IE 5.5* and *6.0*, all concerning object caching. At the time of writing, *Microsoft* has not responded with new patches. Meanwhile, security consultancy *PivX Solutions* lists a running total of 32 unpatched vulnerabilities (http://www.pivx.com/larholm/unpatched/). As ever, though, the problem doesn't stop once patches are provided. A recent poll on the *Virus Bulletin* website asked visitors whether they update their systems frequently with the latest security patches. One would like to assume that visitors to *VB*'s website are at least a little interested in securing their PCs. While 69% said yes, 31% of participants said they do not regularly apply security patches to their systems ▮

With Friends Like These …

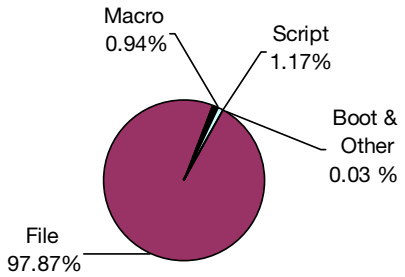
A nuisance email which is neither viral nor a hoax is proving to be equally bothersome. Email recipients are invited to pick up an 'e-card' from the FriendGreetings.com website. Here, the user is asked to install an ActiveX control. Prior to installation, two end-user licence agreements (EULA) are displayed, the second of which states that, by installing the ActiveX control, the user is giving permission to send a similar greetings card to *all addresses* in the user's *Outlook* address book. Unsurprisingly, few users pay sufficient attention to the lengthy EULA and simply agree to the installation of the ActiveX control. With friends like these, who needs enemies? ▮

Prevalence Table – September 2002

Virus	Type	Incidents	Reports
Win32/Klez	File	4987	70.94%
Win32/SirCam	File	815	11.59%
Win32/Yaha	File	448	6.37%
Win32/Magistr	File	207	2.94%
Win32/Nimda	File	92	1.31%
Win32/BadTrans	File	70	1.00%
Win32/Hybris	File	65	0.92%
Redlof	Script	45	0.64%
Win32/Frethem	File	39	0.55%
Laroux	Macro	30	0.43%
Win32/Higuy	File	28	0.40%
Win95/CIH	File	24	0.34%
Win32/Duni	File	22	0.31%
Win32/Elkern	File	18	0.26%
LoveLetter	Script	17	0.24%
Win32/Bugbear	File	11	0.16%
Haptime	Script	9	0.13%
Win32/Funlove	File	9	0.13%
Win95/Tecata	File	9	0.13%
Kak	Script	7	0.10%
Win32/Fbound	File	7	0.10%
Win32/MTX	File	7	0.10%
Divi	Macro	6	0.09%
Win32/Onamu	File	6	0.09%
Others [1]		52	0.57%
Total		7030	100%

[1] The Prevalence Table includes a total of 52 reports across 26 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

Distribution of virus types in reports



CONFERENCE REPORT

A Storm in a Coffee Cup

Helen Martin

The first time VB2002 was dubbed (jokingly) 'the conference from hell' by its organisers, no one imagined how apt that description would turn out to be. However, amongst storms, floods and wrangles with US customs, good humour prevailed, and a thoroughly entertaining conference was enjoyed by all.

Lessons in Meteorology

The *VB* team arrived in New Orleans a couple of days ahead of the start of the conference, to be greeted by an unnerving warning from the staff of the Hyatt Regency. We were warned that evacuation of the hotel (and indeed the entire city) might be necessary, in preparation for another visitor to Louisiana: Hurricane Isadore – expected to arrive on the opening morning of VB2002.

The prospect of 300 delegates each bringing a blanket and a pillow to the hotel ballroom (as indicated in the Hyatt's safety instructions 'what to do in the event of a hurricane') conjured up images of a giant sleepover and memories of the Girl Guides. Perhaps we would have to turn the event into a one-stream conference and convert other hotel guests into AV experts – a good opportunity for end user education perhaps.

Before long, all eyes became glued to *The Weather Channel* and every member of the *VB* crew seemed to have become an expert in hurricane tracking overnight. The outlook seemed gloomy in more ways than one as our welcome drinks reception on board the Creole Queen paddlewheeler had to be cancelled (something to do with not knowing where the Mississippi ended and the streets of New Orleans began), our photographer felt he would be unable to reach the hotel, the entertainment agency admitted it was unlikely any entertainers would make it to the gala dinner, and reports were flying (*ahem*) around the hotel that the airport was closed and delegates would be unable to reach New Orleans.

Meanwhile, the hotel staff were either sent home to batten down their hatches or moved into the hotel, and the hotel entrance was sandbagged as the water level in the loading bay rose to waist height.

As New Orleans began to shut down – shops closed, bars boarded up their windows and sandbagged their doors – the *VB* crew could only feel sorry for those delegates who had come early to do some sightseeing, but who were left with no alternative on the entertainment front than to frequent the hotel bar or channel hop between *The Weather Channel* and the local news stations.

Happily, many delegates had arrived in New Orleans prior to the closing of the airport. As the city prepared for the onslaught of the storm, VB2002 delegates revelled in the 'wartime spirit' and a lively drinks reception was held in the Hyttops sports bar in the hotel – somewhat safer and drier than the Creole Queen under the circumstances.

Thankfully, Isadore had quietened to tropical storm status by the time it reached the Hyatt and, although the hotel slipped a list of safety instructions under each of the bedroom doors, and a city-wide curfew from 10pm to 6am put paid to any late night exploration, it was with relief that we awoke on the opening morning of the conference to find no windows had been blown in, the flood waters had not reached the third floor and the bar was still operational. (Concerned delegates were heard to ask: 'If we get stranded in this hotel, will there be enough wine and beer to keep us going for the duration?')

I Still Dream of Memphis

Yet inclement weather was not the last of the troubles to beset the *VB* crew. While the storm clouds rolled in, another problem was brewing.

A week prior to the conference, a shipment was sent from *VB*'s offices to New Orleans. Boxes full of copies of the conference proceedings, CDs, bags, t-shirts, posters, and so on were packed up carefully, the accompanying paperwork was completed – making sure the contents were itemised in fine detail (or so we thought) – and the shipment was despatched. It made it as far as Memphis, Tennessee.

Apparently, US Customs' idea of fine detail was a little different from *VB*'s – questions fired back at us included 'what is the country of origin of the t-shirts?', 'what ink was used to print the logos on the t-shirts?' and 'how many stitches are there per inch of t-shirt?'. It took more than a little rooting around, but we managed to come up with an answer for every probing question.

However, when it came to coffee cups intended for the sole purpose of pencil-holding at the *VB* registration desk, the matter was passed over to the FDA (Food and Drug Administration). OK, so we were attempting to import two coffee cups emblazoned with the words 'VIRUS BULLETIN', but *we meant computer virus* …

To our horror (and subsequent amusement) the FDA deemed it necessary to send the two coffee cups for laboratory analysis – presumably to be poked, prodded and

tested for new strains of biological virus. Meanwhile, conference registration opened and delegates could be seen wandering around with the perplexed 'I'm sure I'm supposed to have something else' look after being presented with nothing more than a name badge for their efforts.

To our immense relief and gratitude, delegates took the lack of the conference materials in good humour. For the duration of the conference (and despite countless telephone calls attempting to expedite its onward journey), the shipment of proceedings remained in Memphis.

Late on the final day of the conference we were delighted to be informed that some palettes had arrived at the hotel, marked for VB2002. However, it was to our dismay (but, let's face it, not complete surprise) that we discovered that only *half* of the shipment had arrived – and the copies of the proceedings were … still in Memphis. The delegates patiently formed an orderly queue during the afternoon's coffee break while they waited to collect their VB2002 bags and t-shirts, better late than never.

Conference Programme

The disruption caused to travel arrangements by Isadore called for some emergency re-jigging of the programme to allow for two full days of presentations. Reserve speaker Martin Overton stepped in with his paper 'When Worlds Collide' and the ever-versatile Graham Cluley magic-ed up a second presentation ('Viruses: a year in review') to complete the programme.

In his scheduled presentation, Graham Cluley spoke on the subject of e-bugs and debated the question of whether anti-virus vendors should include detection of governments' keystroke logging devices or turn a blind eye. In the absence of an attendee from the FBI, unsuspecting *VB* first-timer Nina Gaubert of the UK's National High Tech Crime Unit was hauled from the audience to take part in a telephone role play with Graham. Game for a laugh, Nina took the challenge in her stride and pulled off an admirable performance – we await VB2003 to see how she chooses to get her own back. Graham's presentation ended with a straw poll indicating that approximately 100% of the audience felt they would like AV vendors to include detection of e-bugs in their products.

Robert Vibert gave a few delegates a rude awakening by catapulting doughnuts into the unsuspecting audience. Prior to the food fight he had been charting the growth of AVIEN since its inception at the *VB* conference in Orlando two years ago and detailing the progress of some of AVIEN's ongoing projects.

On introducing Nick FitzGerald's presentation, session chair Randy Abrams delighted the audience by producing a bottle of Arrogant Bastard Ale, which he explained was intended as an incentive for Nick to, well, get to the point. Nick spoke on the subject of free anti-virus techniques – or was it second-hand cars? After several slides of classic Mini Coopers and Lamborghinis Nick finally did start talking on the subject of anti-virus and justly earned his bottle of ale.

Security consultant Klas Schöldström focused on education, presenting an overview of what he considers to be one of the most effective corporate education tools – a live virus demonstration. Klas detailed how he takes with him a portable virus lab when he visits companies and gave the audience an example of one of his training demonstrations using samples of W32/Explorezip.

John Lambert presented an overview of the Software Restriction Policies feature in *Windows XP* and *.NET Server 2003*, explaining the thinking behind the feature and its use in an anti-malware role. Heuristics came under the spotlight when Andreas Marx reported the results of his retrospective testing of old AV products with newer viruses and Markus Schmall considered the question of whether *Java 2 ME* will provide a playground for malicious code, concluding that, although a secure platform in its current form, mobile telephone vendors, telephone carriers and the AV industry must work together to ensure it remains that way.

A lively panel session brought the conference to a close, with speakers Jeannette Jarvis, Andreas Marx, Richard Marko, Dmitry Gryaznov, John Morris, Carey Nachenberg and (non-speaker) Righard Zwienenberg, who kindly stepped in at the last minute. We learnt that Vesselin Bontchev recently patched his mother's PC for the first time since 1998 and had some interesting discussions including a debate about the likelihood (or otherwise) of cyberattacks being sufficiently severe to bring down the Internet.

The Frills and Spills

An evening of spectacular entertainment at the gala dinner went quite some way towards making up for the disappointment of being unable to cruise the Mississippi on board the Creole Queen paddlewheeler.

The evening kicked off with a fabulous Voodoo show, complete with boa constrictor (although I'm not sure whether delegates were entertained or merely terrified at the point at which the snake was taken on a tour around the audience). The organisers were touched that the Voodoo priestess gave the conference a (much needed) special blessing – we needed all the help we could get!

The evening was rounded off with a rousing performance from a traditional marching jazz band complete with colourful mardi gras revellers – and, for one delegate at least, it was a night to remember as *VB* fixed it for him to fulfil a lifelong ambition to shake Louis Armstrong by the hand.

A sudden and dramatic change in the weather (the pavements were so dry it was hard to believe it had even been raining) enabled delegates to venture outside the hotel after the gala dinner, and most chose to sample the delights (or otherwise) of Bourbon Street, resulting in one or two bleary eyes the following morning (the suggestion that, at future *VB* conferences, Friday's programme reverts to the later start time of 10am has been taken on board – whoever changed it to 9am must have had a sick sense of humour … or an impressive capacity for burning the midnight oil!).

The final day of the conference provided the opportunity for delegates to give us their feedback on the event. All comments have been read and digested. We would like to apologise to the delegate who was 'disappointed that the hurricane was not impressive enough' and sympathise with the delegate whose comment was 'some speakers tend to make you sleep.'

Frequent Flyer

The saga of the missing conference proceedings continued long after the close of the conference. It was with a collective but premature sigh of relief that *VB* received news that the shipment of proceedings had made it back onto British soil. Prepared for a day of boxing, packing, label sticking and mailing materials out to delegates we waited for the shipment to arrive at our office. However, it seemed to have developed a touch of wanderlust. Having touched down in the UK, and before it had even made it out of the airport, the shipment was on a plane flying *back to Memphis* (no, we couldn't believe it either).

VB2003 – Come if you Dare!

While we cannot guarantee fine weather, *VB*'s research team has been hard at work and can confirm that Toronto does *not* lie in a hurricane zone (and for those cautious of *VB*'s recent run of bad luck, or nervous of the fact that next year's will be *VB*'s thirteenth conference, nor does the city lie on an earthquake fault line and there are no active volcanoes in the vicinity). VB2003 will take place at The Fairmont Royal York – itself a Toronto landmark – on 25 and 26 September, 2003. We look forward to seeing you there.

CONFERENCE PREVIEW

AVAR 2002

Allan Dyer, AVAR President

The fifth Association of Anti-Virus Asia Researchers (AVAR) International Conference will be held in Seoul, Korea, on 21 and 22 November 2002.

Several of this year's speeches are reports from government agencies in the region. Seung-Cheol Goh of the Korea Information Security Agency, Zhang Jian of the China National Computer Virus Emergency Response Center, and Shigeru Ishii of the Information technology Promotion Agency in Japan will each report from their region. Meanwhile, Costin Raiu will present an outsider's view of anti-virus protection in Asia using statistics from his Smallpot project. Larry Bridwell will add data from America and show how the malware problem is out of control.

Takuya Yamazaki of the Ministry of Economy, Trade, and Industry in Japan will present the most current concept and views of information security policy in Japan. Jimmy Kuo will review the year, and Paul Ducklin will tell us how to avoid being victims. Looking at blended threats, Motoaki Yamamura will provide a big picture view of what computer systems managers need to know in order to stay ahead of emerging viruses and hacking techniques. Alex Shipp will discuss the different strategies useful for desktop and Internet level anti-virus protection. Randy Abrams will describe and demonstrate the automated virus-scanning system in use at *Microsoft*, while Jong Purisima will address the increasing problem of malware that targets systems as a whole and the future of system disinfection.

Virus naming has long been a controversial topic. Taking the rose by the thorns, Nick FitzGerald will make the first public presentation of the revised *CARO* naming convention. Vesselin Bontchev and Katrin Tocheva will discuss the future of macro and script polymorphism, and Won-Hyok Choi will discuss how to detect and repair viruses that hook the Windows API and attack Win32. Turning this around, Yoshihiro Yasuda will consider the appropriate Win32 hooking that can be used for malware analysis and the design of research tools. SiHaeng Cho will discuss the influence of double-byte character sets in script viruses and worms, and how to prevent this from influencing the integrity of anti-virus software. Finally, Myles Jordan will discuss metamorphism and the application of a meta-heuristic system to detect this viral technique.

The event also features a panel discussion, banquet, the AVAR AGM and a hospitality programme.

Conference:	AVAR 2002, 21–22 November 2002
Venue:	Ritz-Carlton Hotel, Seoul
Web:	http://www.aavar.org/avar2002/index.html

VIRUS ANALYSIS 1

Let free(dom) Ring!

Frédéric Perriot and Péter Ször
Symantec Security Response, USA

On 30 July, 2002 a security advisory from *A.L. Digital Ltd* and *The Bunker* disclosed four critical vulnerabilities in the *OpenSSL* package. *OpenSSL* is a free implementation of the Secure Socket Layer protocol used to secure network communications and it provides cryptographic primitives to many popular software packages, including the *Apache* web server. Less than two months later, the Linux/Slapper worm successfully exploited one of the buffer overflows described in the advisory and, in a matter of days, spread to thousands of machines around the world.

Linux/Slapper is one of the most significant outbreaks on *Linux* systems to date. Although the worm has the potential to infect many more machines, it skips private network classes such as 192.168.0.0/16 intentionally and thus it will not spread on local networks. Slapper shows many similarities with the FreeBSD/Scalper worm, hence the name.

Under Attack

Linux/Slapper spreads to *Linux* machines by exploiting the overlong SSL2 key argument buffer overflow in the libssl library that is used by the mod_ssl module of *Apache 1.3* web servers. When attacking a machine, the worm attempts to fingerprint the system by sending an invalid GET request to the http port (port 80), anticipating that *Apache* will return its version number as well as the *Linux* distribution it was compiled on, along with an error status.

The worm contains a hard-coded list of 23 architectures upon which it was tested and compares the returned version number against this list. It uses this version information later to tune the attack parameters. If *Apache* is configured not to return its version number or the version is unknown to the worm, it will select a default architecture (*Apache 1.3.23* on RedHat) and the 'magic' value associated with it.

This 'magic' value is very important for the worm and is the address of the GOT (Global Offset Table) entry of the free() library function. GOT entries of ELF files are the equivalent of IAT (Import Address Table) entries of *Windows* PE files. They hold the addresses of the library functions to call. The address of each function is placed into the GOT entries when the system loader maps the image for execution. Slapper's aim is to hijack the free() library function calls in order to run its own shell code on the remote machine.

The Buffer Overflow

In the past, some worms have exploited stack-based buffer overflows. Stack-based overflows are the low-hanging fruits compared to second-generation overflows exploiting heap structures. Since the *OpenSSL* vulnerability affected a heap-allocated structure, the worm's author had to deal with a lot of minor details in order to get the attack right for most systems. Thus, exploitation of the vulnerability was not a trivial process.

When *Apache* is compiled and configured to use SSL it listens on port https (port 443). Slapper opens a connection to this port and initiates an SSLv2 handshake. It sends a client 'hello' message advertising eight different ciphers (although the worm supports only one, namely RC4 128-bit with MD5) and gets the server's certificate in reply. Then it sends the client master key and the key argument, specifying a key argument length greater than the maximum allowed SSL_MAX_KEY_ARG_LENGTH (8 bytes).

When the packet data is parsed in the get_client_master_key() function of libssl on the server, the code does no boundary check on the key argument length and copies it to a fixed-length buffer key_arg[] of size SSL_MAX_KEY_ARG_LENGTH, in a heap-allocated SSL_SESSION structure. Thus anything following key_arg[] can be overwritten with arbitrary bytes. This includes both the elements after key_arg[] in the SSL_SESSION structure and the heap management data following the memory block containing the structure.

The manipulation of the elements in the SSL_SESSION structure is crucial to the success of the buffer overflow. The author of the exploit took great care to overwrite these fields in a way that does not affect the SSL handshake very much.

Double-take

Interestingly, instead of using this overflow mechanism just once, the worm uses it twice, first to locate the heap in the *Apache* process address space, and again to inject its attack buffer and shell code. There are two good reasons for splitting the exploit into two phases.

First, the attack buffer must contain the absolute address of the shell code, which is hardly predictable across all servers because the shell code is placed in memory allocated dynamically on the heap. To overcome this problem the worm causes the server to leak the address where the shell code will end up and then sends an attack buffer patched accordingly.

The second reason is that the exploit necessitates overwriting the cipher field of the SSL_SESSION structure located after the unchecked key_arg[] buffer. This field identifies the cipher to use during the secure communication and if its value were lost the session would come to an end too quickly. So the worm collects the value of this field

during the first phase and then injects it back at the right location in the SSL_SESSION structure during the second phase.

This two-phased approach requires two separate connections to the server and succeeds only because *Apache 1.3* is a process-based server (as opposed to a thread-based server). The children spawned by *Apache* to handle the two successive connections will inherit the same heap layout from their parent process. Thus, all other things being equal, the structures allocated on the heap will end up at the same addresses during both connections.

This assumes that two fresh 'identical twin' processes are spawned by *Apache* to handle the two connections. However, under normal conditions, this may not always be the case because *Apache* maintains a pool of servers already running, waiting for requests to handle. In order to force *Apache* to create two fresh processes, the worm exhausts *Apache*'s pool of servers before attacking by opening a succession of 20 connections at 100-millisecond intervals.

The first use of the buffer overflow by the worm causes *OpenSSL* to reveal the location of the heap. It does this by overflowing the key_arg[] buffer by 56 bytes, up to the session_id_length field in the SSL_SESSION structure. The session_id_length describes the length of the 32-byte-long session_id[] buffer located after it in the SSL_SESSION structure. The worm overwrites the session_id_length with the value 0x70 (112). Then the SSL conversation continues normally until the worm sends a 'client finished' message to the server, indicating it wants to terminate the connection.

Upon receipt of the 'client finished' message, the server replies with a 'server finished' message including the session_id[] data. Once again, no boundary check is performed on the session_id_length and the server sends not only the content of the session_id[] buffer but the whole 112 bytes of the SSL_SESSION structure starting at session_id[]. Among other things this includes a field called 'ciphers' that points to the structure allocated on the heap right after the SSL_SESSION structure, where the shell code will go, and a field called 'cipher' that identifies the encryption method to use.

The worm extracts the two heap addresses from the session_id data received from the server and places them in its attack buffer. The TCP port of the attacker's end of the connection is also patched into the attack buffer for the shell code to use later. The worm then performs the second SSL handshake and triggers the buffer overflow again.

Abusing glibc

The second use of the buffer overflow is much more subtle than the first. It can be seen as three steps leading to the execution of the shell code: (1) corrupting the heap management data, (2) abusing the free() library call to patch an arbitrary dword in memory, which is going to be the GOT entry of free() itself, and (3) causing free() to be called again, this time to redirect control to the shell code location.

The attack buffer used in the second overflow is composed of three parts: the items to be placed in the SSL_SESSION structure after the key_arg[] buffer, 24 bytes of specially crafted data, and 124 bytes of shell code.

When the buffer overflow takes place, all members of the SSL_SESSION structure after the key_arg[] buffer are overwritten. The numeric fields are filled with 'A' bytes and the pointer fields are set to NULL, except the cipher field which is restored to the value that was leaked during the first phase.

The 24 bytes of memory following the SSL_SESSION structure are overwritten with fake heap management data. The glibc allocation routines maintain so-called 'boundary tags' in between memory blocks for management purposes. Each tag consists of the sizes of the memory blocks before and after it plus one bit indicating whether the block before it is in use or available (the PREV_IN_USE bit). Additionally, free blocks are kept in doubly linked lists formed by forward and backward pointers maintained in the free blocks themselves.

The fake heap management data injected by the worm after the SSL_SESSION structure poses as a minimal-sized unallocated block, containing just the forward and backward pointers set respectively to the address of the GOT entry of free() minus 12 and the address of the shell code. The address of the GOT entry is the 'magic' value determined by fingerprinting, and the address of the shell code is the value of the ciphers field leaked by *OpenSSL* in the first phase of the attack, plus 16 to account for the size of the fake block content and trailing boundary tag.

Once these conditions have been set up on the server, the worm sends a 'client finished' message specifying a bogus connection id. This causes the server to abort the session and attempt to free the memory associated with it. The SSL_SESSION_free() function of the OpenSSL library is invoked and this in turn calls the glibc free() function with a pointer to the modified SSL_SESSION structure as an argument.

One might think that freeing memory is a simple task. In fact, considerable book-keeping is performed by free() when a memory block is released. Among other tasks free() takes care of consolidating blocks, merging contiguous free blocks into one to avoid fragmentation. The consolidation operation uses the forward and backward pointers to manipulate the linked lists of free blocks, and trusts these to be pointing to heap memory (at least in the release build).

The exploit takes advantage of the forward consolidation of the SSL_SESSION memory block with the fake block created after it by setting the PREV_IN_USE bits of the boundary tags appropriately. The forward pointer in the fake block which points to the GOT is treated as a pointer to a block header, dereferenced, and the value of the backward pointer (the shell code address) is written to

offset 12 of the header. Thus the shell code address ends up in the GOT entry of free().

It is worth noting that the fake backward pointer is also dereferenced, so the beginning of the shell code is treated as a block header, and patched at offset 8 with the value of the fake forward pointer. To avoid corruption of the shell code during this operation, the shell code will start with a short jump followed by ten unused bytes filled with NOP-s.

Finally, on the next call to free() by the server, the modified address in the GOT entry of free() is used and the control flow is directed to the shell code.

Shell Code and Infection

When the shell code is executed it searches first for the socket of the TCP connection with the attacking machine. It does this by cycling through all file descriptors and issuing a getpeername() call on each until the call succeeds and indicates that the peer TCP port is the one that was patched into the shell code. Then it duplicates the socket descriptor to the standard input, output and error.

Next it attempts to gain root privilege by calling setresuid() with UIDs all set to zero. *Apache* usually starts running as root and then switches to the identity of an unprivileged user 'apache' using the setuid() function. Thus the setresuid() call will fail because, unlike the seteuid() function, setuid() is irreversible. The author of the shell code appears to have overlooked this fact, but the worm does not need root privileges to spread, since it writes only to the /tmp folder.

Finally, a standard shell '/bin/sh' is executed with an execve() system call. A few shell commands are issued by the attacker worm to upload itself to the server in uuencoded form, and to decode, compile and execute itself. The recompilation of the source on various platforms makes the identification of the worm in binary form a little more difficult. The operations are carreid out in the /tmp folder where the worm files reside under the names .uubugtraq, .bugtraq.c and .bugtraq (notice the leading dots to hide the files from a simple 'ls' command).

Now you see me, Now you don't!

Since the worm hijacks an SSL connection to send itself, it is legitimate to wonder whether it travels on the network in encrypted form. This question is particularly crucial for authors of IDS systems that rely on detecting signatures in raw packets.

Fortunately, the buffer overflow occurs early enough in the SSL handshake, before the socket is used in encrypted mode, thus the attack buffer and the shell code are clear on the wire. Later, the same socket is used to transmit the shell commands and the uuencoded worm also in plain text. The 'server verify', 'client finished' and 'server finished' packets are the only encrypted traffic but they are not particularly relevant for detection purposes.

P2P

When an instance of the worm is executed on a new machine it binds to port 2002/UDP and becomes part of a peer-to-peer network. Notice that, although a vulnerable machine can be hit multiple times and exploited again, the binding to port 2002 prevents multiple copies of the worm from running at the same time.

The parent of the worm (on the attacking machine) sends to its offspring the list of all hosts on the peer-to-peer network and broadcasts the address of the new instance worm to the network. Then periodic updates to the hosts list are exchanged between the machines on the network. The new instance of the worm also starts scanning the network for other vulnerable machines, sweeping randomly chosen Class B-sized networks.

The protocol used in the peer-to-peer network is built on top of UDP and provides reliability through the use of checksums, sequence numbers and acknowledgement packets. The code has been taken from an earlier tool and each worm instance acts as a DDoS agent and a backdoor. Much has been written on this topic so we won't repeat information that is available elsewhere.

Conclusion

Linux/Slapper is an interesting patchwork of a DDoS agent, some functions taken straight from the OpenSSL source code and a shell code the author says is not his own. All this glued together results in a fair amount of code, not easy to figure out rapidly. Like FreeBSD/Scalper, most of the worm was probably already written when the exploit became available and, for the author, it was just a matter of integrating the exploit as an independent component.

And, as in Scalper, which exploited the BSD memcpy() implementation, the target of the exploit is not just an application but a combination of an application and the runtime library underneath it. One would expect memcpy() and free() to behave a certain way, consistent with one's everyday-programming experience. But when used in an unusual state or passed invalid parameters, they behave erratically.

Linux/Slapper shows that *Linux* machines can become the target of widespread worms just as easily as *Windows* machines do. For those with Slapper-infected *Linux* servers this will be a day to remember.

Linux/Slapper	
Alias:	Apache_mod_ssl.
Type:	C sources based worm that compiles itself to ELF; performs DDoS attacks; spreads via buffer overflow attacks against vulnerable versions of OpenSSL.

VIRUS ANALYSIS 2

You've Got M(1**)a(D)i(L+K)l

Peter Ferrie
Symantec Security Response, Australia

Encryption techniques have evolved over the years, from simple bit-flipping, through polymorphism, to metamorphism, and combinations of these have been used as well (for example, see *VB*, May 2002, p.4). All of these techniques have one thing in common: they are applied to the virus body. The alternative is to apply them to the thing that contains the virus body. This variant of the Chiton family, which the virus author calls W32/Junkmail, is one of those.

To the Manner Born

When Junkmail is started for the first time, it decompresses and drops a standalone executable file that contains only the virus code, using a 'fixed' (taking into account the variable name of the Windows directory) filename and directory. As with the other viruses in the family, Junkmail is aware of the techniques that are used against viruses that drop files, and will work around all of the counter-measures: if a file exists already, then its read-only attribute (if any) will be removed, and the file will be deleted. If a directory exists instead, then it will be renamed to a random name. The structure of the dropped file is the same as that used by W32/Gemini (see *VB*, September 2002, p.4) and W32/EfishNC (Junkmail is based very heavily on that virus). If the standalone copy is not running already, then Junkmail will run it now. The name of the dropped file is 'Exp*I*orer.exe'. Depending on the font, the uppercase 'i' will resemble a lowercase 'L', making the viral process difficult to see in the task list.

Hook, Line, Sinker

After dropping the standalone copy, Junkmail will alter the Registry in such a way that the virus is run whenever an application is launched. Junkmail alters the 'Shell\Open\Command' keys for the 'com', 'exe', and 'pif' extensions in both the 'LocalMachine' and 'CurrentUser' hives. Both hives are altered because in *Windows 2000* and *XP*, the Current User values override the Local Machine values. The three extensions are altered because they are all associated with applications. Additionally, the change makes removal more difficult because if the virus is removed before the Registry is restored, then applications cannot be launched easily. Fortunately, some improvisation allows for ways around this problem.

If the computer is running *Windows NT/2000/XP*, then the virus will add itself as a service. The virus does not start the service, perhaps because the standalone copy is running already, and *Windows* will perform that action anyway, when the computer is rebooted. If the computer is running *Windows 9x/ME*, then the virus will place an undocumented value in an undocumented structure, which results in the task not being displayed in the task list. This mimics the actions of the undocumented RegisterServiceProcess() API.

It Takes Two to Argue

Whenever the standalone copy is executed, the virus will parse the command-line to determine why it is running. The parsing is done in the platform-independent way that is favoured by the virus author – if the computer is running *Windows 9x/ME*, then the virus will use the ANSI APIs to examine characters; if the computer is running *Windows NT/2000/XP*, then the virus will use the Unicode APIs to examine characters. If there are arguments on the command-line, then the virus assumes that it was launched via the Registry alteration, and will attempt to execute the application that is named in the first argument.

If there are no arguments on the command-line, then the virus assumes that it has been launched as the standalone copy, and will execute its main code. The main code begins by retrieving the addresses of the APIs that it requires and creating the threads that will allow the virus to perform several actions simultaneously.

Threads

The first thread runs once every hour. It will enumerate all drive letters from A: to Z:, looking for fixed and remote drives. If such a drive is found, then the virus will search in all subdirectories for files to infect. Files will be infected if they are *Windows* Portable Executable files for *Intel* 386+ CPUs, and are not DLLs.

The method of infection is the same as for some other variants in the family – the virus will either append its data to the last section, or insert its data before the relocation table, and alter the entrypoint to point directly to the virus code. For files that do not possess the infection criteria, the suffix of their name is checked against a list of files that might contain email addresses. The virus is interested in files whose suffix is 'asp', 'cfm', 'css', or 'jsp', or contains 'php' or 'htm'. If such a suffix is found, then the file is searched for a 'mailto:' string, and the email address that follows is saved for later.

The second thread runs once every two hours. It will enumerate the network shares and attempt to connect to them. If the connection succeeds, then the virus will search in all subdirectories for files to infect.

The third thread also runs once every two hours. It will attempt to connect to random IP addresses. There are two routines for this action, one for ANSI platforms, and one for

Unicode platforms. If the connection succeeds, then the virus will search in all subdirectories for files to infect.

The fourth thread is the one from which the virus gains its name. It runs once after every six hours, and will send a single email to the last address that the virus found while searching for files to infect. It is also here that the encryption is applied to the container, rather than the virus body. The virus sends itself using the MIME message format, as described in RFC 1521. While this should present no problems, it appears that a number of developers have overlooked one significant sentence: 'All header fields defined in this document, including MIME-Version, Content-type, etc., are subject to the general syntactic rules for header fields specified in RFC 822. In particular, all can include comments'. The result is that an email that would normally look like this:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary=TFICLMGJ
```

can be altered to look like this:

```
M(F)IM(])E-(*/
*)V(y)e(7)r(*)s(U*0)i(*LZ)o(H)n(.):(l)
1(:*=).0
Content-Type: mul(26)t(fH*)ip(|*)a(***)rt(*)/
mi(/*j)x(8)e('M)d;
 (<|)bo(*,)u(1**)nda(D)r(L+K)y=TFICLMGJ
```

In case that wasn't bad enough, the virus contains an abundance of other tricks – the subject is chosen randomly from a list, or in some cases will contain only a random filename and no other text. The message body contains variable parts, so one message body could begin with:

```
I received this file from you yesterday
evening.

I think it was sent without you knowing by
the Badtrans trojan.

The filename was altered but it looked like
an important document inside.
```

while another could begin with:

```
I received this file from you yesterday
morning.

I think it was sent without you knowing by
the Sircam worm.

The filename was changed but it looked like
an important picture inside.
```

I've Been Framed

The attachment type is chosen randomly from a list. Some of the types are those that are vulnerable to the IFrame exploit that allows automatic execution of the attachment. There are 22 of these types. The other types are those that will display the CID instead of the filename when prompting the user to open or save the attachment. The CID has been named 'email' with this in mind – the person who views the message will see a prompt to Open or Save an attachment called 'email', and will likely select Open. There are four of these types.

The filename of the attachment is also 'email', followed by one or two suffixes, chosen randomly from lists. Some viruses use '.bat' as a suffix even though the file is binary, however Junkmail uses the suffix in the correct way – if .bat is chosen, the virus sends itself as a real .bat file. If .shs is chosen, the virus sends itself as an OLE2 file. Otherwise, the virus sends itself in the *Windows* PE file format.

Layer upon Layer

The .bat method is an interesting technical achievement. There are certain characters that are interpreted differently on *Windows 9x/ME* and *Windows NT/2000/XP*. The virus author is aware of this, and the .bat code is able to determine the *Windows* platform and allow for the differences. Following the platform determination is a line containing executable code composed entirely of printable characters. The technique is known as 'executable ASCII'. The code is only 217 bytes long, but it is able to decode a base64 attachment, write it to a file, then launch that file. The decoder itself is only 59 bytes long. The rest of the .bat file is the base64-encoded copy of the virus. If the .bat file is executed, it will determine the *Windows* platform, create a temporary file and write both the decoder and attachment there, then run the temporary file. The temporary file will decode and run the attachment, which will launch the virus.

The structure of the OLE2 file is not constant either, thanks to a feature of *Windows*. The file contains only the absolute minimum number of components required to run – one storage and one stream. When the file is executed, *Windows* will automatically create the 'missing' storages and streams and update the file structure, resulting in a file that could be several times its original size.

Conclusion

The RFCs are full of features that many people might, but very few people do, use. This can lead to complacency among developers, leading to loopholes, leading to Junkmail and those that will follow it. Engine developers need to re-read the RFCs and implement support for even the most obscure features because, as is demonstrated here, these unusual features can be used for unusual purposes.

W32/Junkmail

Alias:	W32/Chiton variant.
Type:	Memory-resident parasitic appender/inserter, slow mailer.
Infects:	*Windows* Portable Executable files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.

RESEARCH PROJECT

Malformed Email Project

Andreas Marx

At the end of 2001, a rapidly increasing number of email worms were using malformed emails to spread. Popular mail clients, such as *Outlook* and *Outlook Express*, are perfectly able to decode damaged or invalid messages containing attachments. However, we realised that a lot of content security programs, such as email virus scanners, were not scanning such attachments at all – because they were not RFC-compliant.

RFC stands for 'Request for Comments' – a set of technical and organizational notes about the Internet which cover many aspects of computer networking, and many of which represent Internet standards, either by practical use or by agreement. For example, they explain how SMTP (Simple Mail Transfer Protocol) or MIME (Multipurpose Internet Mail Extensions) must be implemented and how they work, so that software based on these standards is interoperable. The RFCs can be found at http://www.ietf.org/rfc.html.

Early in 2002 email security problems attracted the interest of the security community. Many methods by which a content scanner can be bypassed were published, yet still many security programs were unable to find attachments in messages whose formatting was a little out of the ordinary.

As partial fixes, anti-virus companies added detection for known viruses using this method as they were transferred as EML files (in RFC 822 format). However, without analysing the problem properly and trying to fix it in their SMTP/MIME parser, any subsequent viruses using the same vulnerabilities to hide themselves would not be detected.

It was reasonable to think that there may be more problems which were as yet undiscovered. A little investigation and experimenting showed that there were indeed several more ways in which a virus could get past email scanners.

During February and March 2002 we (Andreas Marx and Mark Ackermans) discussed possible ways in which these known and a lot of unknown email scanner vulnerabilities could be solved in mail content security software within an acceptable length of time. It was from these discussions that the idea of the malformed email project came about. We enlisted the support of *Virus Bulletin* and embarked on the project.

The Test

A test set was created by Mark Ackermans, based heavily on the eicar.com file – at the time of writing (October 2002), this includes about 370 samples which 'hide' attachments, trick scanners or cause buffer overflows which can be used for DoS attacks, for example. However, it includes only email structure vulnerabilities and no other mail-related security issues, such as script exploits. Meanwhile we compiled a list of developers at AV/security companies who needed to be notified – currently this list stands at 89 companies.

On 3 April 2002, most of the AV/security companies on our list were sent an email (see next page), setting out our findings and indicating what we proposed to do to fix the problem (some companies were notified a little later, to allow us to answer incoming messages first and send out the test set, under non-disclosure).

We explained that the purpose of the test set is to stop malware in malformed messages – all files which can be decoded to malware by commonly used mail clients should be detected by mail security products. When malformed message parts are detected, blocking or removing the malformed part of the message is acceptable, although preference is given to virus detection, especially if blocking causes false positives and when blocking can be disabled or is disabled by default.

We thought that two months should be an acceptable length of time for companies to test their own products and to address possible issues, after which time they were to send the fixed products to us for testing. However, due to a very high number of requests for the test set, we chose to extend the deadline by one and a half months – until 22 July 2002. At the time of writing we are still receiving products for testing. (Of course, those products that failed to meet our deadline will be marked clearly as such in the final test results.)

To date, we have collected 270 different products from 43 companies for final testing. The products will be tested in the next few weeks, mainly by *AV-Test.org*'s Marc Schneider, who is working on this project as part of his diploma thesis.

Of course, for testing purposes, a new test set will be used in place of the original version distributed to the companies. This will contain real viruses instead of eicar.com, with similar and published security exploits, in order to ensure that the problems have, indeed, been fixed.

Initial Contact

The following is a copy of the first standard email we sent out to all anti-virus and content security companies, as well as other developers whose products are likely to be affected by these vulnerabilities:

```
Hello!
We've found out that your products are
```

very likely vulnerable to a few MIME and UUEncode problems, which makes attachments "invisible" for your product (e.g. not filtered or scanned), but well-known email programs, will "see" the attachments. For example, Outlook Express uses very liberal decoding - it is able to decode a lot of the malformed attachments correctly, therefore we used OE 5.5 SP2 for our internal tests.

"We" are Andreas Marx, team leader at the Anti-Virus Test Center at the University of Magdeburg, Germany (http://www.av-test.org/) - performing tests for more than 30 international publications; Helen Martin, Editor of Virus Bulletin, England (http://www.virusbtn.com/) and Mark Ackermans, The Netherlands. Other anti-virus organisations are also involved.

A few of these problems are already known about and have been published without informing the vendors first (e.g. http://www.security.nnov.ru/advisories/content.asp and Bugtraq postings), others are as yet still unknown. Currently, we have about 300 malformed MIME and UUEncode files in our collection and a relatively small nymber of these anomalies are publicly known at this time. We have not tested all files with your product, due to the high number of available products, but your solution is at least vulnerable to a few tested attacks.

The good point is that some of these data are 'too malformed' to be recognised as valid attachments - such files should be stopped by your solution as being an invalid file (some will be stopped, some not). However, a lot of the rest will get through your product fully unscanned and unfiltered, which is indeed a very risky issue. A few viruses are already known which use such malformed attachment encoding - mostly, because of bugs (e.g. Win32/Badtrans.B, Win32/Sircam.A, Win32/FBound.C).

Another good example is Win32/Gibe.A: It inserts spaces in front of the base encoding which seems to result in corruptions by decoders (http://vil.nai.com/vil/content/v_99377.htm). Most programs (for example, Outlook and WinZip) won't properly decrypt the first line and therefore the worm does not work any more (corrupted sample), but a few programs like Outlook Express will see and decrypt the attachment correctly. The worm is fully able to work.

Another example to by-pass a few programs are too long file names or a "." at the end of the filename (e.g. "test.exe."), even if this was not the main topic we've been working on (see http://online.securityfocus.com/archive/1/265387 for details).

We're almost sure that other virus authors and hackers will find out more of our ways to "bypass" your product, too. Therefore, we have decided to send out this advisory - to you and other affected vendors. A few vulnerabilities have already been known for quite a long time, but they are not yet fixed. The CERT/CC is informed as well, but currently too busy, so we've decided to inform you now and not when it is too late.

I hope, we can work together to fix these issues. In this case, and if you agree not to make our test samples and demo scripts and information available to third parties, we'll send you a password-protected or PGP'ed RAR archive with all of the sample files, descriptions of the test set and tools needed to test your product. We will also assist you and answer your questions (please contact Mark with CC: to me).

During the time in which we've prepared the test set, more and more problems were made publicly available. Therefore, we can only suggest that you work carefully on these issues. We think, a good timeframe would be one month to test your product, identify and fix these problems and another one for final testing, if everything is running fine. After these two months, we'll collect all products from the informed vendors and test if all of these holes are closed. These final test results will be published in an upcoming Virus Bulletin issue and on our web site AV-Test.org. No exclusions - all products will be listed and tested. ;-)

I'm looking forward to your feedback.

Andreas Marx

What Happens Next?

The intention is to publish all test results in *Virus Bulletin*, starting in the February 2003 issue. Any review or claim made prior to this publication date that a company's product detects all samples in the test set used in this project, or that it can detect all malformed emails, cannot be verified.

Furthermore, it should be noted that detection of all files in the test set does not guarantee that the program is completely safe – the test set does not contain examples of all known mail security problems.

It is possible that we have missed a few companies or that some developers may have overlooked our initial warning in April and the reminders in the following months. If you are a developer working on content security or related products, and you are interested in gaining access to the test set and further documentation, please contact us (email editor@virusbtn.com). For security reasons, the list of companies that have been notified and those who responded cannot be published at this point.

FEATURE

Fishing for Hoaxes: Part 2

Pete Sergeant

In the first part of this feature (see *VB*, October 2002, p.13) I looked briefly at why virus hoaxes are a problem, why it would be useful to set up an automated method of detecting hoaxes, and discussed some possible approaches, before deciding on Bayesian classification. This month I shall explain the three tools I wrote to help me do this, and give some examples that demonstrate how effectively these programs worked, as well as looking at the weaknesses of the Bayesian approach, and at some other interesting information that can be deduced from the data.

The Programs

The first program is fairly unremarkable – it takes the different formats in which I have been given sample data, strips out headers, strips out some signatures, and dumps the body of the email into a file.

Ideally, if these were all emails I had received personally, I would maintain the headers, and use them to aid classification – emails with the word 'AVIEN' in their headers almost certainly shouldn't be classified as hoaxes, for example. However, most of those who supplied me with sample data took some steps to obscure the identity of the original senders, and strip headers, with varying degrees of success – I needed to standardize if my efforts were to achieve any degree of success.

The Trainer

The second of my programs is the trainer – this goes through my email samples, and allocates each word a percentage chance that the email in which it appears is a hoax. There are two main steps involved here.

First, all our sample emails need to be split up into individual words. In the name of simplicity, a word is defined as a string containing only any of the characters from a to z – any other characters are regarded as being word-separators. Words of fewer than three characters are dropped, as most are of little value – orphaned letters from abbreviations, prepositions, and so on.

Each time we find a word, the counter for that word is incrememented in the appropriate category (hoax or non-hoax). We also keep track of how many hoaxes and non-hoaxes we are using for training.

Next, using this data, we define the probability that an email containing the word in question is a hoax.

The code for this looks something like the following:

```
sub get_prob {

  my $word = shift;

  my ($ngood, $nbad) = ($big_data{mails}->{0},
      $big_data{mails}->{1});
  my $g = $big_data{words}->{$word}->[0];
  my $b = $big_data{words}->{$word}->[1];

  return undef unless ($g + $b) > 5;

  my $prob = min(1.0, $b/$nbad) / (min(1.0, $g/
     $ngood) + min(1.0, $b/$nbad));

  return 0.99 if $prob > 0.99;
  return 0.01 if $prob < 0.01;
  return $prob;

}
```

There are two interesting things to note here, both of which come from Paul Graham's original Lisp code for catching spam using a similar technique.

First, if there are fewer than five occurrences of a word in the entire training set, it's probably not a good indicator either way, so we ditch the word. Secondly, we stop any words from having absolute probabilities – we don't have a sufficiently large training set to say that any word is *definitely* an indicator either way, so we fudge it a little.

The Classifier

The third program is the classifier itself. This takes an email, tokenizes it into words (as explained above), picks the 15 most 'interesting' words (those whose value is furthest from the neutral 0.5), and uses Bayes theory to work out the probability from these 15 probabilities that the email is a hoax.

Running the Classifier on Test Sets

So, how effective is this method? There are two ways to get an idea of the classifier's effectiveness. First, are any of my clean sample set (non-hoaxes) classified as hoaxes, and are any of my dirty set (hoaxes) classified as not being hoaxes?

Clearly, this is not the fairest method of analysis, but it's quite interesting anyway: no hoaxes were identified as

being non-hoaxes, and 0.01% of clean emails were identified as being hoaxes. Closer inspection of those emails that were misclassified showed, with one exception, that they were all 'throwaway' three- or four-word emails, of little value unless read in the thread from which they were taken. Boring. But let's look at the genuinely misclassified email:

```
–
I received this a few minutes ago from one of
my users. It seems that several users have
seen this in the last few hours. Comments?

--

Subject: Fw: Rotary cards

WARNING!!! IF YOU RECEIVED THIS E-MAIL FROM
ME DO NOT OPEN IT. DELETE IT IMMEDIATELY
THIS E-MAIL WAS SENT TO ME AND THE ATTACHMENT
(Rotary Card.doc.com) IS A VIRUS. Don't open
it. Delete immediately.
–
```

The filename it mentions certainly looks like something you wouldn't want to touch – but then, we really don't know. If we can't classify this ourselves, how can we expect a computer to do so?

Let's look at what, exactly, the classifier took offence to (note, percentages have been rounded):

```
bash-2.05a$ ./rate.pl test_sets/nothoaxes/1419
Probability: 0.95
cards – 0.01
comments – 0.01
card – 0.98
delete – 0.94
delete – 0.94
warning – 0.90
open – 0.89
open – 0.89
seen – 0.11
users – 0.15
users – 0.15
attachment – 0.15
immediately – 0.80
immediately – 0.80
minutes – 0.75
```

The word 'card' will be penalized heavily due to the 'A Virtual Card For You' hoax; the words 'immediately' and 'delete' are part of the typical hoax call to action; the only surprises for me are that the word 'minutes' is penalized so heavily (perhaps because Virus X will 'destroy your hard drive in minutes'?), and that 'attachment' gets off so lightly … the moral is that it's not always the words you might expect that have the biggest impact.

Some Real World Examples

Running the program on our test sets is bound to give good results, so it's time to see how well the method fares against real-world examples.

The easiest way to do this for hoaxes is to remove some hoaxes from the training set, and then to scan against them. The hoaxes picked (pretty much at random) were WTC survivor, and jdbgmgr.exe.

Google returned *Symantec*'s description of WTC Survivor as its top link, so this is the version against which I ran the test. The results were fairly conclusive:

```
bash-2.05a$ ./rate.pl dirty/wtc_survivor.txt
Probability: 0.99
survivor – 0.01
survivor – 0.01
survivor – 0.01
libraries – 0.99
erase – 0.98
dynamic – 0.96
dear – 0.95
warned – 0.96
book – 0.95
delete – 0.94
receive – 0.94
receive – 0.94
everyone – 0.94
drive – 0.93
forward – 0.93
```

Initially, I was a little surprised that the phrase 'dynamic link libraries' was punished so harshly – 99 per cent of the occurrences of the word 'libraries' in my test bodies were in hoaxes. However, on grep'ing through my test sets, I realised quickly that libraries are mentioned in the 'A Virtual Card For You' hoax. The fact that hoax writers, like virus writers, are often keen to recycle really helps out here.

The only other surprise was the penalization of the word 'dear' – then again, my clean sample set is almost entirely composed of anti-virus email, rather than personal email: I'm forced to conclude that those in the anti-virus industry are an unfriendly bunch :-).

However, jdbgmgr.exe proved to be quite a bugbear, as the initial variant was both fairly 'well' written and original. Essentially, the hoax contained a number of words considered, in our test set, to be clean, and these overruled the 'dirtier' words.

So what does this tell us about the effectiveness of the technique? For rehashed and recycled hoaxes, where large sections are borrowed from previous ones, it does very well, and it also does very well at identifying hoaxes we've seen before, and picking them from virus-related but non-hoax email.

However, given something that introduces a lot of new vocabulary (like 'messenger'), it doesn't perform so well. Therefore, to work in the real world, you'd want to keep the classifier well-trained – a lot like anti-virus software: good heuristics work a lot of the time, but fall down when something very different is introduced.

Rogues' Gallery

In the first part of this article, I identified some words which, I anticipated, would have high hoax ratings – in particular, names of anti-virus vendors.

So who's top of the bunch? The anti-virus vendors are somewhat interesting, and one vendor really seems to be leading the pack:

mcaffe – 0.99

norton – 0.88

mcaffee – 0.78

mcafee – 0.70

It's worth noting that these aren't absolute figures. What this does show is that hoax writers have real difficulty in spelling *McAfee*, and that, in comparison to its occurrence in non-hoax but AV-related emails, *McAfee* features very heavily in hoaxes.

This also demonstrates a strength of the Bayesian approach – it probably wouldn't have occurred to me (for a while anyway) to look for misspellings and penalise them.

Speed

Is this system fast enough to use in the real world? Well that really depends on how you write it. My early attempts managed only a measly two hoaxes per second, but once I set my mind to removing some unnecessary bottlenecks, and wrote a daemon in Perl to do my scanning, my throughput was roughly 30 hoaxes a second – not bad for a Perl script on a Mac G3.

Considering that most emails wouldn't have to be touched, and if one was setting this up for a production environment it's likely that some rather beefier hardware would be thrown at it, and it would be written it in a lower-level language, I was impressed.

Conclusion

The system isn't perfect, and maybe we don't receive a sufficient number of hoaxes yet to run the risk of misclassification, but it is interesting to see that we can achieve some pretty accurate results.

Of course, it would have been nice to have used an even larger test set, as well as a large corpus of non-virus-related email, but time constraints were a limiting factor with this. (My thanks to those who were kind enough to provide me with samples.)

The question left in my mind is how well this technique would work for email-borne viruses – certain mime types and file extensions would be likely to be heavily penalized, as would certain keywords (I certainly don't tend to receive any non-viral emails that contain the word 'spicegirls' for example …).

COMPARATIVE REVIEW

Windows 2000 Advanced Server

Matt Ham

This has been a year for new platforms for the VB100% award, with AV products for *Linux* and *Windows XP* already having been submitted to the trials and tribulations of the test procedures. On this occasion the test platform is less novel, yet still untested in its server version: *Windows 2000 Advanced Server.*

This server product is not radically different from the venerable *Windows NT Server*, and the problems encountered with the products on test were (in most cases) overcome easily, being the same as those encountered many times before on the older platform.

Also this month, a potential long-term problem vanished from the test sets. After, by virus standards, an eternity in the In the Wild (ItW) test set, Michelangelo finally dropped out of this month's test. This was the last boot sector virus in the set to have a file system which appears corrupt to most, if not all, *Windows* installations, and for this reason was more difficult to detect for some products.

As for additions to the test set, all but four were what are these days the usual suspects, the *Windows*-executable-based worm. A notable newcomer, in type if not difficulty of detection, was BAT/Hitout.A. This is the first batch virus to be In the Wild since BAT/911, and thus potentially could have caused problems due to extension. However, these problems did not materialise in practice.

Further Clarifications

At regular intervals discussions arise as to exactly what a VB 100% award actually means. Although developers are generally aware of the exact relevance, it appears that some end-users have been examining the figures in a manner which somewhat distorts the meaning of the award. Another frequent question we are asked by readers is 'which product is best?', which falls into a related category.

A VB 100% award denotes that the product tested showed, in its default mode, 100 per cent detection of In the Wild test samples and no false positives in a selection of clean files. For on-demand scanning of files, detection is considered to be a note in the product log file that the file is infected or very likely so. For on-demand scanning of boot sector viruses, a notification or log file entry is required.

For on-access scanning the matter is a little more confusing, since the best method of testing – executing all files and using the results from this activity – is clearly impractical.

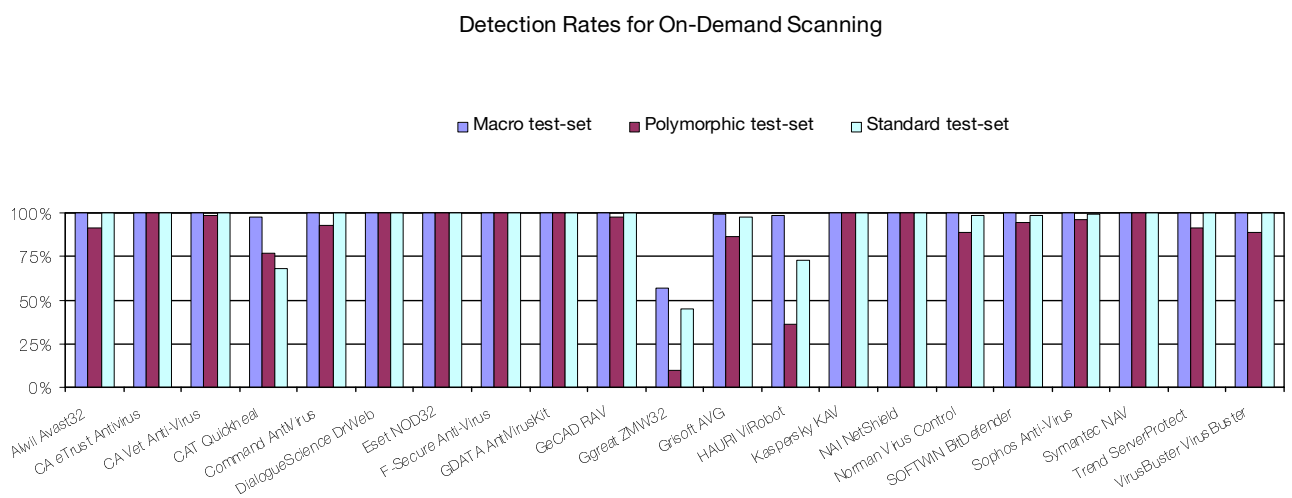
On-access tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
Alwil Avast32	0	100.00%	0	100.00%	100.00%	14	99.66%	144	91.13%	13	99.73%
CA eTrust Antivirus	0	100.00%	0	100.00%	100.00%	4	99.90%	0	100.00%	0	100.00%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	436	98.58%	4	99.78%
CAT Quickheal	0	100.00%	0	100.00%	100.00%	110	97.25%	3774	76.85%	613	67.68%
Command AntiVirus	0	100.00%	0	100.00%	100.00%	0	100.00%	133	93.02%	12	99.61%
DialogueScience DrWeb	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.85%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.85%
GDATA AntiVirusKit	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
GeCAD RAV	0	100.00%	0	100.00%	100.00%	0	100.00%	35	97.61%	2	99.88%
Ggreat ZMW32	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Grisoft AVG	2	99.58%	0	100.00%	99.60%	20	99.51%	251	86.05%	59	97.65%
HAURI ViRobot	1	99.83%	0	100.00%	99.84%	69	98.19%	10695	35.96%	N/A	N/A
Kaspersky KAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
NAI NetShield	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	7	99.49%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	0	100.00%	650	88.92%	31	98.58%
SOFTWIN BitDefender	0	100.00%	11	0.00%	94.74%	0	100.00%	126	94.73%	2	99.88%
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	11	99.73%	60	95.79%	18	99.42%
Symantec NAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	13	99.87%
Trend ServerProtect	9	98.94%	0	100.00%	99.00%	0	100.00%	292	91.15%	8	99.82%
VirusBuster VirusBuster	0	100.00%	8	27.27%	96.17%	49	98.96%	160	89.13%	11	99.67%

Detection is thus judged by a product denying access to an infected file when the file is opened for writing.

For boot sector on-access scanning a visible notification or log file entry is required. In this case denial of access is not a useful guide to detection since the *VB* boot sector test floppies are all blank as far as file contents are concerned. Since denial of access is likely to show a blank disk as the only detectable effect, this is not particularly useful. The addition of extra files to the disk for use in deciding whether access has been denied was decided against, for in past testing some products were only able to detect a boot sector virus on a floppy containing other files – a situation which would be apparent only with the use of disks in their current state.

There have been products which, by design, do not scan on access except on file execution. Thankfully, those that are designed this way becoming fewer overall. More problematic are those products which cannot be cajoled into producing reasonable logs on demand, thus making detection checking problematic. These are checked by setting the product to delete and/or disinfect. The files are then scanned until no more detections are present, if necessary manually noting those files which are detected as infected but are not deleted or disinfected. Disinfected files are removed from the test set by use of CRC checking, and those files left in the test set are considered to be misses.

This said, there remains ample opportunity for products to miss detection, in our tests, of files which they are perfectly



able to detect – which begs the question, why should this be so? The answers are potentially many, though two are more relevant than others. First, there are the matters of default extension lists, a common area for failure over the years. In particular *Kaspersky Anti-Virus* and *NAI* products have failed to gain a number of VB 100% awards because the default extension lists did not include possible extensions for In the Wild viruses. In most cases these extension-based problems are easily solved by an administrator adding extensions to the default list. We could perform these changes prior to testing. We feel, however, that our readers are better served if they know that they have to do this, than if we scan all files regardless of extension.

Another example of why some products miss out on VB 100% awards, is where certain files are not scanned directly on-access. The usual assumption by the product developers is that the files will be scanned when passed on to an application which makes use of them. At the most common level this covers such objects as ZIP files, which are often not scanned until unzipped. In some past tests *Aladdin*'s products fell into this category where OLE files were concerned, scanning these only when passed to, for example, *Word*. The most recent example of this behaviour has been the *FRISK* treatment of EML files, which are not scanned until individual mails are pulled from within (see this issue, p.3). From a developer's point of view these choices make sense in that leaving objects unscanned until use creates fewer overheads. The chance of infection on a protected machine is not increased, since scanning will occur before code execution.

Such treatment of objects does, however lead to misses under the VB 100% testing methodology, which brings us back to the original questions. In short, the answers are as follows. A VB 100% award means that a product has passed our tests, no more and no less. The failure to attain a VB 100% award is not a declaration that a product cannot provide adequate protection in the real world if administered by a professional. As to which product is 'best', this all depends on the interaction between the anti-virus software, installed hardware and software and that same administrator. We would urge any potential customer, when looking at the VB 100% record of any software, not simply to consider passes and fails, but to read the small print in these reviews.

Alwil Avast32 3.0.499.2

ItW Overall	100.00%	Macro	99.66%
ItW Overall (o/a)	100.00%	Standard	99.73%
ItW File	100.00%	Polymorphic	91.13%

As ever, misses in detection were scattered through the non-ItW sets with a definite favouring of the polymorphic set for non-detection. The *Alwil* interface is one of the more complex and customisable of those on offer, though it seemed that on-access scanning had become simpler to configure. This might have been as a result of finding a control already in existence, though unseen before. Whatever the reasons, the testing ran smoothly. With no false positives and full detection In the Wild, Avast 32 chalks up the first VB 100% of this review.

Cat Computer Services QuickHeal X Gen 6.05

ItW Overall	100.00%	Macro	97.25%
ItW Overall (o/a)	100.00%	Standard	67.68%
ItW File	100.00%	Polymorphic	76.85%

Another relative newcomer to the comparative scene, *Quickheal* showed improvements in detection. From a historic-virus viewpoint detection remains weak in some areas, though as more modern threats are considered the detection rate improves rapidly. Speed of scanning is good too, which leaves only the matter of false positives as a possible fly in the ointment. Again this is an area where rapid improvements have occurred, and a lack of false positives and a full detection of In the Wild Viruses gains *QuickHeal* a VB 100% award.

On-demand tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
Alwil Avast32	0	100.00%	0	100.00%	100.00%	14	99.66%	144	91.13%	13	99.73%
CA eTrust Antivirus	0	100.00%	0	100.00%	100.00%	4	99.90%	0	100.00%	0	100.00%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	436	98.58%	2	99.90%
CAT Quickheal	0	100.00%	0	100.00%	100.00%	110	97.25%	3774	76.85%	613	67.68%
Command AntiVirus	0	100.00%	0	100.00%	100.00%	0	100.00%	133	93.02%	10	99.73%
DialogueScience DrWeb	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	1	99.98%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	1	99.98%
GDATA AntiVirusKit	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
GeCAD RAV	0	100.00%	0	100.00%	100.00%	0	100.00%	35	97.61%	2	99.88%
Ggreat ZMW32	274	54.80%	0	100.00%	57.18%	1805	56.46%	14772	9.84%	1056	45.08%
Grisoft AVG	2	99.58%	0	100.00%	99.60%	20	99.51%	251	86.05%	59	97.65%
HAURI ViRobot	1	99.83%	0	100.00%	99.84%	69	98.19%	10695	35.96%	541	72.80%
Kaspersky KAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
NAI NetShield	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	4	99.63%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	0	100.00%	650	88.92%	29	98.71%
SOFTWIN BitDefender	1	99.92%	0	100.00%	99.92%	14	99.64%	121	94.76%	47	98.30%
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	8	99.80%	60	95.79%	18	99.42%
Symantec NAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	13	99.87%
Trend ServerProtect	9	98.94%	0	100.00%	99.00%	0	100.00%	292	91.15%	8	99.82%
VirusBuster VirusBuster	0	100.00%	0	100.00%	100.00%	0	100.00%	160	89.13%	8	99.82%

Command AntiVirus for Windows 4.73.1

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.73%
ItW File	100.00%	Polymorphic	93.02%

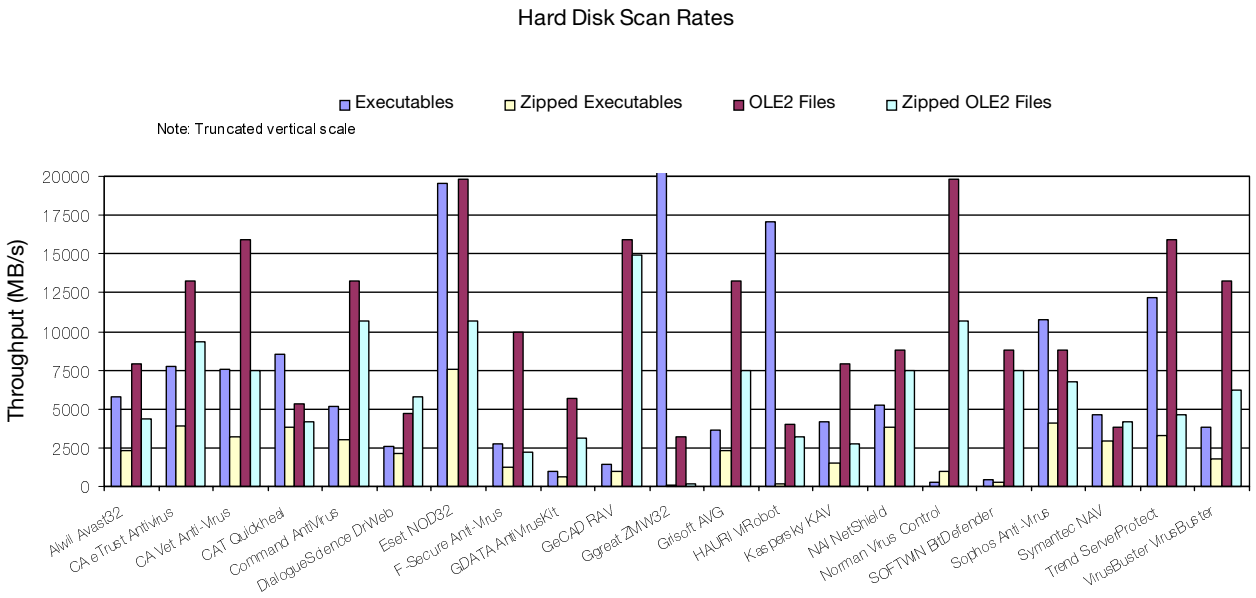
Historically, *Command AntiVirus* has been a pleasure to review, with easy installation, operation and log file analysis. Now, however, log files are by default, produced in RTF format – which rendered useless the standard file comparison tools used in log analysis. The hidden RTF content more than doubled the size of the report file as compared with a plain text version of the same data. These irritations aside, *Command AntiVirus* earned a VB 100%.

Misses were mostly among the modern W32 polymorphics, notably W32/Fosforo, W32/Etap, W32/Tuareg.B and W32/Zmist.D. There were also some small floppy change detection problems when testing these on access.

Computer Associates eTrust Antivirus 6.0.96 23.57.56

ItW Overall	100.00%	Macro	99.90%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItW File	100.00%	Polymorphic	100.00%

As is becoming traditional for *eTrust*, a selection of mandatory patches needed to be applied on installation.



However, these were more automated than I remember. Not quite so good is the slightly confusing labelling of updates on the *CA* site. Also, the update instructions fail to note that manual halting of services within *eTrust* is required before patching can occur. After installation, however, results were of the customary high standard. Even the commonly missed samples of W32/Etap were detected, though samples of W97M/Box.A were missed. Fortunately these were no barrier to *eTrust Antivirus* earning a VB 100%. With reference to the comments made earlier in this review, W32/Heidi.A samples embedded within zip files were detected on demand, though not on access.

Computer Associates Vet Anti-Virus 10.52.02

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.90%
ItW File	100.00%	Polymorphic	98.58%

Vet is less burdened or blessed (depending upon user needs) with integration into other *Computer Associates* products than *eTrust Antivirus*, which adds to its simplicity of use in this kind of test. The age-old cry of weakest in the polymorphics goes up yet again – with detection elsewhere being all but perfect. Remaining quite speedy on scanning, and with no false positives, *Vet* earns *Computer Associates* another VB 100% award.

DialogueScience DrWeb for Windows 95-XP 4.28c

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.98%
ItW File	100.00%	Polymorphic	100.00%

DrWeb has had a history of good results in *VB* comparative testing, which has also accompanied a gradual improvement in interface clarity for the on-access component. The number of suspicious files on this occasion was only one. All but one detected sample in the test set were detected exactly without recourse to heuristic methods, leaving only ZIP-encoded W32/Heidi.A files on access and the TMP sample of W32/Nimda.A as misses. Since the latter is included only as a curiosity in the standard set, *DrWeb* gains another VB 100% award.

Eset NOD32 1.314

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItWmFile	100.00%	Polymorphic	100.00%

NOD32 remained speedy, but was rivalled on this occasion by other products. As far as detection was concerned, full In the Wild detection for both boot and file viruses was sufficient to garner another VB 100% award for the product. Misses were, in fact, absent in any test set.

F-Secure Anti-Virus 5.40

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.98%
ItW File	100.00%	Polymorphic	100.00%

Sporting a combination of two engines, neither of which are poor at detection, it comes as no surprise that *FSAV* collects another VB 100% award. As is a common theme in this review, the ZIP files containing W32/Heidi.A were the only misses of any note.

Hard Disk Scan Rate	Executables			OLE Files			Zipped Executables		Zipped OLE Files	
	Time (s)	Throughput (MB/s)	FPs [susp]	Time(s)	Throughput (MB/s)	FPs [susp]	Time (s)	Throughput (MB/s)	Time(s)	Throughput (MB/s)
Alwil Avast32	94	5818.4		10	7933.4		69	2310.4	17	4388.7
CA eTrust Antivirus	71	7703.3		6	13222.3		41	3888.2	8	9325.9
CA Vet Anti-Virus	72	7596.3		5	15866.8		50	3188.3	10	7460.7
CAT Quickheal	64	8545.8		15	5288.9		42	3795.6	18	4144.9
Command AntiVirus	107	5111.5		6	13222.3		52	3065.7	7	10658.2
DialogueScience DrWeb	214	2555.8	[1]	17	4666.7		75	2125.6	13	5739.0
Eset NOD32	28	19533.3		4	19833.4		21	7591.3	7	10658.2
F-Secure Anti-Virus	201	2721.1		8	9916.7		130	1226.3	33	2260.8
GDATA AntiVirusKit	585	934.9	1	14	5666.7		244	653.3	24	3108.6
GeCAD RAV	377	1450.7		5	15866.8		166	960.3	5	14921.5
Ggreat ZMW32	18	30385.1	4	25	3173.4		2068	77.1	413	180.6
Grisoft AVG	150	3646.2	[5]	6	13222.3		70	2277.4	10	7460.7
HAURI ViRobot	32	17091.6	[1]	20	3966.7		928	171.8	23	3243.8
Kaspersky KAV	132	4143.4		10	7933.4		104	1532.9	27	2763.2
NAI NetShield	104	5259.0		9	8814.9		42	3795.6	10	7460.7
Norman Virus Control	1792	305.2		4	19833.4		167	954.6	7	10658.2
SOFTWIN BitDefender	1156	473.1	1	9	8814.9		549	290.4	10	7460.7
Sophos Anti-Virus	51	10724.2		9	8814.9		39	4087.6	11	6782.5
Symantec NAV	118	4635.0		21	3777.8		55	2898.5	18	4144.9
Trend ServerProtect	45	12154.0		5	15866.8		49	3253.4	16	4663.0
VirusBuster VirusBuster	142	3851.6		6	13222.3		88	1811.6	12	6217.3

GData AntiVirusKit Professional 11.0.4

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItW File	100.00%	Polymorphic	100.00%

The second multiple-scan-engine product in this review and again the policy seems to have paid off. In *GData*'s case the result is a total absence of missed files in any test set. However, a false positive for *AVK* can be blamed for its failure to carry off the VB 100% award on this occasion.

GeCAD RAV AntiVirus Desktop 8.6.103

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.88%
ItW File	100.00%	Polymorphic	97.61%

Like many of the other products this month, *RAV* passed through the review process without hiccups. W32/Heidi in its ZIP archive was missed both on access and on demand, together with W32/Etap, one sample of W32/Fosforo and some more surprising samples of Cryptor. None of these were In the Wild, however, and *RAV* gains a VB 100% award.

Ggreat ZMW32 virus scan M7.5+

ItW Overall	57.18%	Macro	56.46%
ItW Overall (o/a)	N/A	Standard	45.08%
ItW File	54.80%	Polymorphic	9.84%

Ggreat's product is new to the *VB* comparative tests, and enters at a slight disadvantage by nature of its design. Primarily, it is a scanner for incoming emails, and as such,

has no other on-access portion. This disqualifies it from a VB 100% award. As far as detection overall was concerned, selection of directories seemed to have unpredictable results as to how many were scanned, so the scanning was performed in areas rather than the whole collection in one batch. Stability problems were encountered when repair was selected.

GriSoft AVG 6.0 build 398

ItW Overall	99.60%	Macro	99.51%
ItW Overall (o/a)	99.60%	Standard	97.65%
ItW File	99.58%	Polymorphic	86.05%

AVG brought a slight need for the use of judgement to the definition of default mode, since it offers three scan types as existing options from its main scan interface. Of Quick, Complete and Main, Main was selected as the default scan type on the basis of its name.

HAURI ViRobot 4.0

ItW Overall	99.84%	Macro	98.19%
ItW Overall (o/a)	99.84%	Standard	72.80%
ItW File	99.83%	Polymorphic	35.96%

ViRobot has been one of the recent marked improvers in performance in VB 100% testing, and this review showed increased detection again. The log files available in the product are, however, still limited in size to such an extent that they are not useful for testing purposes. Detection was judged here by deletion of some infected files and disinfection of others, followed by deletion of those files with an altered CRC. Of note in this process was W32/Beast which, in its DOC samples, was flagged as being removable only upon the next boot.

A rather larger problem was encountered when the standard set was scanned on access. On several samples in this set the machine would reproducibly blue-screen with an error which looks likely to be related to unpleasant pitfalls within these samples. The Standard test set was eventually listed as untested on access due to time constraints.

Kaspersky Anti-Virus 4.0.5.35

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	100.00%
ItW File	100.00%	Polymorphic	100.00%

Kaspersky Anti-Virus was for a long period a scanner that could do no wrong in *VB* comparatives, though suffering from a hiatus mainly brought about by extension issues. These issues seem to have been dealt a serious and happy fatal blow. Misses in the test set were completely absent, as were false positives. *Kaspersky Anti-Virus* earns another VB 100% award.

NAI NetShield 4.5 4.1.60 4.0.4227

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.63%
ItW File	100.00%	Polymorphic	100.00%

Another a product which has suffered from issues with extensions in its recent history, and another which seems to have overcome these lately. Although the optional all-file scanning patch was not applied, its status being definitely not a patch which is required to keep the product up to date, this did not harm the results in any way. Misses were confined to the common W32/Heidi on-access and to this Cruncher was added – another virus which encodes itself, in this case using DIET. *NetShield* produced no false positives and thus a VB 100% award is awarded.

Norman Virus Control 5.4

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	98.71%
ItW File	100.00%	Polymorphic	88.92%

Norman remains unique in its method of scan job construction, a feature which has ceased to be a novelty when reviewing. One major change that has occurred is that the product once again produces log files without recourse to undocumented features.

The slowdown on the *VB* clean executable test remains all the more strange because it is not reflected when the same files are scanned in zipped format. Weakest on polymorphics but with a clean record on false positives, *NVC* gains a VB 100% award for *Norman*.

SOFTWIN BitDefender Professional 6.4.3

ItW Overall	99.92%	Macro	96.64%
ItW Overall (o/a)	94.74%	Standard	98.30%
ItW File	99.92%	Polymorphic	94.76%

BitDefender continues to show reasonable detection rates in all sets, though missing out on some scattered samples, most notably amongst the polymorphics. A single miss of the HTM sample of W32/Nimda.A In the Wild, however, was sufficient to deny the product a VB 100% award. This was most likely due to choices in the implementation of on-demand scanning, since the same file was detected on access.

Matters were more clear cut when it came to problems in the on-access boot sector testing. During these tests no alerts were triggered at any time. Similarly, no detection was logged by the various statistical methods on offer for examining scan results, and this test set thus drew an effective blank as far as detection was concerned.

Sophos Anti-Virus 3.62

ItW Overall	100.00%	Macro	99.80%
ItW Overall (o/a)	100.00%	Standard	99.42%
ItW File	100.00%	Polymorphic	95.79%

The *Sophos* product showed a significant cosmetic change, with a whole new corporate image having been impressed upon it. However, the product itself and the majority of the GUI remains the same.

With only superficial changes to the product, scanning matters changed little if at all. Those files missed were those missed by *SAV* since time immemorial (Positron, Navrhar and the like), in addition to a fair number of the newer polymorphic viruses. Since none of these are from the ItW set, *SAV* earns itself another VB 100% award.

Symantec Norton AntiVirus Corporate Edition 8.00.9374

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	100.00%	Standard	99.87%
ItW File	100.00%	Polymorphic	100.00%

Norton AntiVirus showed its usual good rates of detection, though not without some oddities creeping in. BAT/911.A was missed both on access and on demand, and perhaps more oddly, W97M/Antisocial.F was apparently missed only on demand – a virus having previously had perfect detection. This turned out to be due to an odd quirk in logging for this virus which declared non-existent files to be infected, while making no mention of the existing files. Despite this odd behaviour, which was noted as a detection nonetheless, and slow scanning of infected sets, results were otherwise excellent and *NAV* gains another VB 100% award for its pains.

Trend ServerProtect 5.35 1047

ItW Overall	99.00%	Macro	100.00%
ItW Overall (o/a)	99.00%	Standard	99.82%
ItW File	98.94%	Polymorphic	91.15%

Trend's offering suffered from slightly dated virus definitions. A definition update was promised, but did not arrive. Given that the misses that occurred in this test included W32/Surnova.D and W32/Datom.A, recent additions to the In the Wild set, it is likely that this lack of upgrade had an effect upon detection rates. Blame for the lack of a VB 100% award, however, cannot be laid entirely at the foot of this update issue, since an ItW sample of W32/CTX.A was also missed. Other than this, detection was very good except in the polymorphic sets, traditionally a weak spot, and the category under which W32/CTX.A can also be placed.

VirusBuster for Windows 3.10

ItW Overall	100.00%	Macro	100.00%
ItW Overall (o/a)	96.17%	Standard	99.82%
ItW File	100.00%	Polymorphic	89.13%

VirusBuster's initial installation resulted in a blue screen upon the required reboot after installation. However, the reference to a bad pool caller was not reproducible either on this initial installation or on a second installation on a fresh image of the operating system. The latter installation was used for testing, in case the initial blue-screen had left *VirusBuster* in some way defective.

Problems were apparent in the on-access scanning of boot sectors, where change detection was in a league of its own as far as irritation was concerned. In several sessions of testing, and two further reinstallations, three viruses were detected once on access, after which detection seemed not to exist. Given the good results on other scanning, this comes as something of a disappointment.

Other than these on-access woes there was full detection of all but a scattering of polymorphic samples, with no false positives. A close approach to a VB 100% award, scuppered by boot sectors.

Conclusion

The most notable feature of this review, from a practical point of view, was the contrast with the recent *Windows XP* comparative. In that review the problems encountered both on installation and operation were many. In this review the problems were few, far between and by and large reserved for those products less frequently reviewed. This can be ascribed to the additional age of *Windows 2000 Advanced Server* and to its similarity to *Windows NT Server* – both of which factors will have given developers ample time to iron out any odd bugs.

This difference in performance goes a long way towards explaining why many businesses seem to lag far behind the times when it comes to upgrading operating systems. *Windows NT* and *2000* still hold sway in great swathes of the corporate market, and the stability of time-tested software upon them plays a large part in this reluctance to speed on to the newest platform of *XP*. It is not without reason that some users prefer platforms that the manufacturers now decry as being feature-barren, antiquated and due for replacement.

Technical Details

Test environment: Three 1.6 GHz Intel Pentium 4 workstations with 512 MB RAM, 20 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy, all running *Windows 2000 Advanced Server Service Pack 2*.

Virus test sets: Complete listings of the test sets used are at http://www.virusbtn.com/Comparatives/Win2K/2002/test_sets.html.

A complete description of the results calculation protocol is at http://www.virusbtn.com/Comparatives/Win95/199801/protocol.html.

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, 6 Kimball Lane, Suite 400, Lynnfield, MA 01940, USA

Tel (781) 9731266, Fax (781) 9731267

END NOTES AND NEWS

The CSI 29th Annual Computer Security Conference and Exhibition will be held 11–13 November 2002 in Chicago, IL, USA. The conference is aimed at anyone with responsibility for or interest in information and network security. For more information email csi@cmp.com or see http://www.gocsi.com/.

The 5th Anti-Virus Asia Researchers (AVAR) Conference takes place 21–22 November 2002 at the Ritz-Carlton, Seoul, Korea. Topics covered will include information on how the AV community works together globally, the latest virus and AV technologies, and reports on virus prevalence in various countries in Asia. The conference will be hosted by *Ahnlab, Inc*. For more information see http://www.aavar.org/.

Infosecurity 2002 conference and exhibition will be held 10–12 December 2002 at the Jacob K. Javits Center, New York, USA. For further details, including information on exhibiting and conference registration, see http://www.infosecurityevent.com/.

Papers and presentations are being accepted for the Black Hat Windows Security 2003 Briefings. Papers and requests to speak will be received and reviewed until 15 December 2002. The Briefings take place 26–27 February 2003 in Seattle, WA, USA. For details of how to submit a proposal see http://www.blackhat.com/.

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

Infosecurity Italy will be held in Milan, Italy, 12–14 March 2003, for details see http://www.infosecurity.it/.

RSA Conference 2003 takes place 13–17 April 2003 at the Moscone Center, San Francisco, CA, USA. General Sessions feature special keynote addresses, expert panels and discussions of general interest. This year's Expo will feature more than 138,000 square feet of exhibit space with more than 200 vendors. Optional tutorials and immersion training sessions will provide the basics of e-security technology, enterprise security and security development techniques, and 13 class tracks will feature a wide variety of workshops, seminars and talks. See http://www.rsaconference.net/.

Infosecurity Europe 2002 takes place 23–25 April 2003, Olympia, London. A free keynote and seminar programme alongside almost 200 exhibitors is expected to attract more than 7,000 dedicated security visitors. See http://www.infosec.co.uk/.

Information Security World Asia takes place 23–25 April 2003, Suntec Singapore. See http://www.informationsecurityworld.com/2003/iswa_SG/.

Black Hat Europe 2003 takes place 12–15 May 2003 at the Grand Krasnapolsky, Amsterdam, the Netherlands. For more details see http://www.blackhat.com/

As many as 80 per cent of computer users in China may have suffered viral infections on their machines. A report released by China's Ministry of Public Security revealed that 83.98 per cent of participants in a survey carried out by the Ministry had seen their machines hit by viruses – an increase of 10 per cent from last year.

Command Software Systems Inc.* has been merged with authentication platform developer *Authentium, Inc. Helmuth Freericks, CEO *Command Software*, said, 'In today's environment, Internet security companies need to provide a broad range of integrated products designed to combat emerging risks. Our merger with Authentium gives us access to new markets, a broader array of advanced technologies, and top tier partners.' See http://www.commandsoftware.com/.

***Network Associates* has begun distribution of its security update-reminder service**, together with a free trial of its online virus-scanning service, as part of *Microsoft*'s next version of its *MSN* Internet-access service. *Microsoft* will include *McAfee Security Center* on a CD that gets customers started on *MSN 8*, as part of a revenue-sharing pact *Microsoft* signed with *McAfee.com* earlier this year. A free 120-day trial of *McAfee VirusScan Online* will also be included with the new *MSN* version. See http://www.nai.com/.

***Eset* has been ranked among the 50 fastest-growing technology companies in Central Europe** in *Deloitte & Touche*'s Technology Fast 50 list for the year 2002. As if to prove the point, *Eset* has sponsored a hand-built 450kw Mosler MT900R taking part in this month's inaugural Bathurst 24-Hour Endurance Race in Sydney, Australia. For more information see http://www.nod32.com/.