AUGUST 2007

VIRUS BULLETIN

Fighting malware and spam

CONTENTS

IN THIS ISSUE

GOING TO THE DARK SIDE

Dr Vesselin Bontchev shares his views on whitelisting and explains why, contrary to some recent reports, conventional anti-virus scanners will be around for a long time to come.

page 4

BINARIES IN PROFILE

Instrumenting binaries is a technique that is rapidly gaining popularity among security researchers. Profiling the binary prior to instrumentation can provide a lot of important information about the target. Aleksander Czarnowksi has the details.

page 9

VB100 – 64-BIT WINDOWS VISTA

The rough terrain provided by the test platform this month tripped up several products in their quest for VB100 status. John Hawes has the full details.

page 13

vb Spam supplement

This month: anti-spam news & events, and Alberto Treviño and Dr J. J. Ekstrom detail the advantages and shortcomings of sender authentication as an anti-spam technique, explaining why they consider it a useful tool in the fight against spam.

ISSN 1749-7027

virus BULLETIN COMMENT

'Anyone who writes even one example of a piece of malware, exploit or rootkit feels qualified to call himself a security researcher.'

Aleksander Czarnowski
AVET, Poland

ARE YOU INVISIBLE?

Recently the security community has been busy discussing a bet made over the detectability of a rootkit, after Joanna Rutkowska claimed that she and her *Invisible Things* team are able to create a rootkit that is undetectable.

As someone working for a company that offers penetration testing and forensic analysis services among other things, I am very interested in rootkit technology. During pen-testing, rootkits can provide a great risk demonstration after gaining control of a system, so they add value both to the process and the customer. In the case of forensic analysis we need to identify how system security has been compromised and to what extent the attacker has penetrated the system. This means rootkit detection as well. This puts us in an interesting position when sometimes rootkits are bad, and sometimes they are a good thing.

Personally, I think the issue is not as technical as it seems, and is a lot broader than just rootkits. The real problem is the number of researchers who fail to do their research properly before making claims. These days it seems that anyone who writes even one example of a piece of malware, exploit or rootkit feels qualified to call himself a security researcher. However, the reality is that becoming a security researcher takes a lot more than a few minutes or hours of hacking. It involves a lot of research including research into what has happened in the past.

Editor: Helen Martin

Technical Consultant: John Hawes

Technical Editor: Morton Swimmer

Consulting Editors:
Nick FitzGerald, *Independent consultant, NZ*
Ian Whalley, *IBM Research, USA*
Richard Ford, *Florida Institute of Technology, USA*
Edward Wilding, *Data Genetics, UK*

The past is important here because similar claims about 'undetectable'/'unbeatable' malware have been made in the past. None lasted very long. Repeating such claims just makes me wonder about the reasons for doing so. What's more, part of the technology is already well known. Do you know how to bypass all *Windows Vista* anti-rootkit safeguards? Run it within *VMware* – you then have total control of the operating system execution environment. Does *Vista* complain when it runs inside a virtual machine? No. So theoretically somebody could say that he has found a system vulnerability or a rootkit that is undetectable (by the operating system).

Now let's consider the term 'invisible' or 'undetectable'. If I understand these correctly, such a rootkit should always be hidden so that it leaves no sign of its presence. So we could argue that even a simple 'hello world!' message on the screen would make the rootkit visible. If I can see it, I can detect it.

Ms Rutkowska should also prove that her rootkit is 100% bug free and it will never crash any system during operation. We all know that this problem is non-trivial from a mathematical point of view. If the system crashes due to rootkit installation, it will be visible. This is important as Ms Rutkowska's rootkit technology targets a very broad range of modern PCs. The trend among current exploits is that they increasingly target specific systems due to differences and safeguards like address randomization. So it's a very brave approach to try to target a broad range of systems in today's world.

Let me come back to the crash problem for a while. If I can see it (crash) I can detect it, which brings us to the question of 100% detection. Can anyone show me a 100% detection rate without false positives in anti-malware or IDS/IPS solutions? Anyone?

Last, but I guess not least is the issue of money. A newly established company requesting financial support for its research in the way in which Ms Rutkowska and her *Invisible Things* team have done is a bit strange. Does this mean they don't have customers who would back up their research investment? I hope not! When you try to sell something it must be useful. I really can't see any benefits to a customer paying almost half a million dollars for such an experiment, but I'm sure there would be a lot of customers willing to pay half a million for a solution that would provide an organization with some benefits. So it seems that somebody had an interesting idea and certain technical knowledge but no business plan or vision of how to sell it. Does gambling make it more sellable? I'd bet not.

The views presented in this article are the author's own, not those of his employer.

NEWS

AV MARKET GROWTH AND PREDICTIONS

According to analyst firm *Frost & Sullivan*, the worldwide market for anti-virus solutions reached US$4.68 billion in 2006 – an increase of 17.1% from the previous year. The analysts expect the market to grow at approximately 10.9% compound annual growth rate for the next six years, reaching US$9.69 billion by 2013. The firm cites an increase in end-user awareness of targeted online crimes such as identity theft, loss of enterprise data and extortion, as a driving force behind the growth of the anti-malware market.

Meanwhile, in a more localised round of number crunching and statistics bandying, it has been reported by *Anti-Malware.ru* that the Russian anti-virus software market grew by 49% in 2006, reaching $68.6 million. With a small consumer market thanks to the prevalence of software piracy in the region, the main growth areas were in sales of software to corporate and small to medium-sized business customers.

According to the report, *Kaspersky Lab* retains its place as market leader with a 36.3% share of the Russian market, followed by *Symantec* and *Trend Micro*. The Russian-founded *Doctor Web* sits in fourth place, while *Eset* made it into the region's top five for the first time in 2006, with a 292% record growth in the region.

SECURITY FOR CRITICAL INFRASTRUCTURES

The US Department of Homeland Security (DHS) has laid out a set of security requirements for automated control systems to protect the country's critical infrastructure and key resources against online attacks.

The recommendations in the Catalog of Control System Requirements include basic IT security measures such as installing anti-virus software and keeping it fully up to date. The document indicates that, for maximum security, remote updates for security software should be scheduled for periods when the control system is disconnected from the equipment it controls. The document also recommends against using DNS for control systems, in order to protect against denial of service attacks, and against using Voice over IP, Instant Messaging, FTP, HTTP and file sharing on control systems.

Elsewhere the document, which was put together by representatives of the Department of Energy National Laboratories and the National Institute of Standards and Technology, details practices that are recommended to increase physical security, including organisational, personnel and environmental security practices. The full set of recommendations can be seen at http://www.us-cert.gov/control_systems/pdf/COR-07-12332_20070710-final.pdf.

Prevalence Table – June 2007

Virus	Type	Incidents	Reports
W32/Bagle	Worm	2,754,822	27.63%
W32/Netsky	Worm	2,295,459	23.02%
W32/Mytob	Worm	1,916,596	19.22%
W32/MyWife	Worm	759,543	7.62%
W32/Virut	File	402,459	4.04%
W32/Lovgate	Worm	400,842	4.02%
W32/Zafi	Worm	327,747	3.29%
W32/Mydoom	Worm	146,896	1.47%
W32/Bagz	Worm	130,391	1.31%
W32/Stration	Worm	104,530	1.05%
W32/Rontokbro	Worm	101,766	1.02%
W32/Rjump	Worm	73,867	0.74%
W32/VB	Worm	66,783	0.67%
W32/Parite	File	60,134	0.60%
W32/Jeefo	File	54,712	0.55%
VBS/Small	Worm	45,257	0.45%
W32/Funlove	File	42,563	0.43%
W32/Perlovga	Worm	30,197	0.30%
W32/Looked	File	28,613	0.29%
VBS/Butsur	Script	27,391	0.27%
W32/Klez	File	24,855	0.25%
W32/Fujacks	File	22,959	0.23%
W32/Sality	File	17,895	0.18%
W32/Mabutu	Worm	14,294	0.14%
W32/Sohanad	Worm	11,209	0.11%
W32/Tenga	File	11,174	0.11%
W32/Wukill	Worm	11,028	0.11%
W32/Allaple	Worm	8,794	0.09%
W32/Bugbear	Worm	7,400	0.07%
VBS/Redlof	Script	6,609	0.07%
W32/Chir	File	5,557	0.06%
W32/Alman	File	5,132	0.05%
Others[1]		53,282	0.53%
Total		9,970,756	100%

[1]The Prevalence Table includes a total of 53,282 reports across 121 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

OPINION

THE DARK SIDE OF WHITELISTING

Dr Vesselin Bontchev
FRISK Software International, Bulgaria

For the past year and a half, Robin Bloor has repeatedly posted to his blog and to various other places on the web articles about how ‘anti-virus’ is dead or dying slowly, or at least in serious trouble. One of his articles [1] caught my attention. There were so many wrong things in it, that I could not resist posting a comment in response. Later it occurred to me that such articles might mislead some people, so I decided it was worth writing an article on this subject myself – something people could use as a reference for debunking such delusions.

Like many people who are not anti-virus specialists, Mr Bloor understands the term ‘anti-virus’ essentially to mean known-malware scanners. Of course, this is by no means the only kind of anti-virus technology in existence – it is only the most widely used one. In [2] I list seven main types of anti-virus programs, some of which are further subdivided into half a dozen subtypes – and that source is already out of date, having been written more than a decade ago.

The reason why Mr Bloor has become disenchanted with ‘anti-virus’ seems to be because it has become difficult for the known-malware scanners to cope with the constantly increasing flood of new malware. For instance, the Storm family of Trojan horses uses what we call ‘server-side polymorphism’ – every time someone downloads one, the server generates a new variant, resulting in about 54,000 variants released in just one week [3]. Unlike conventional polymorphism found in viruses, this malware is not self-replicating and the polymorphic engine is not present in the malware for the anti-virus researcher to analyse and develop generic detection for. Of course, this does not mean that there are no efficient techniques against this (see [3] for details). Nowadays there are additional techniques like sandboxing, for instance.

Conventional scanners are notoriously bad at coping with non-replicating malware. A virus replicates – so once the AV developer implements detection of it in the scanner, other users will be protected. However, trojans are usually one-shot weapons – at the time at which the anti-virus developers receive a sample, the victim is already compromised and the same trojan is unlikely to be used again against somebody else, so implementing detection of it isn’t of much help to anyone.

Of course, the fact that the number of known malware programs is increasing at an ever faster rate has prompted many people to predict, over the past couple of decades, that at some point conventional scanners will ‘die’ – i.e. will become ineffective. Nevertheless, conventional scanners are still very much alive and kicking – although the technology behind them has improved significantly (albeit in ways not immediately obvious to the user). There are perfectly good reasons for this – conventional scanners are the kind of anti-virus programs with which the average user is most comfortable, in the sense that he or she can easily understand and use them. I am pretty sure that another decade from now conventional scanners will still be in use, and people will still be predicting their imminent demise.

Another thing about conventional scanners that Mr Bloor does not like (and he is certainly not alone in this), is that they require constant updating. That is, the user has to keep paying for them.

Anyway, since Mr Bloor has decided that conventional anti-virus does not work any more, he is left with the question of what to replace it with – because *something* has to be used to protect the user from malware, right? In this case, the panacea seems to be the method known as whitelisting.

WHAT IS WHITELISTING?

If you consider how conventional scanning works, it basically builds a blacklist – a list of known bad programs that shouldn’t be allowed to run on the user’s computer. As new malicious programs appear, they are added to the blacklist – this is why scanners need constant updating.

Whitelisting is the opposite of this. Instead of building a list of things that should *not* be allowed to run, the anti-virus program uses a list of things that *should* be allowed to run – and denies execution of everything that is not on this whitelist.

The idea of whitelisting is certainly not new. For a long time, people have been joking that the number of known malicious programs is increasing so quickly, that at some point it will be easier to scan for known good programs instead. As early as 1990, Dr Fred Cohen [4] proposed the idea of ‘integrity shells’ – programs that would only allow the execution of software with known-good integrity (using a database of checksums to control integrity and relying on the fact that viruses have to modify the programs they infect, thus destroying their integrity). I use the *Kerio* personal firewall in ‘paranoid’ mode, which makes it issue an alert every time an ‘unapproved’ program is executed – this is a kind of whitelisting, too.

Since whitelisting has been around for almost two decades, the natural question to ask is: if it is so much better than conventional scanning, why has it not replaced the latter already? The answer, of course, is because it cannot. It, too, is fraught with problems. On its own, whitelisting cannot stop malware – just as, on its own, conventional scanning (or other ‘blacklisting’-based approaches) cannot.

THE PROBLEMS OF WHITELISTING

Basically, the problems of whitelisting boil down to two issues: *what is a program?*, and *which programs are good?* [5]. (Just like the problems of any blacklisting-based approach boil down to two issues – *what is a program?* and *which programs are bad?*)

Exotic execution exceptions, or what is a program?

Theoretically, the question of ‘what is a program?’ is an unsolvable one. That is, for any given sequence of symbols, there exists at least one Turing Machine for which this sequence is a program, if it appears somewhere on the machine’s tape.

Of course, real computers are not Turing Machines (a Turing Machine has infinite memory, for instance) – but we hit another snag there. All contemporary computers are based on the von Neumann architecture – and an underlying principle of this architecture is the equivalence between code and data. That is, one program’s code is another program’s data and vice versa. For instance, a JavaScript text is a program to the browser – but it is data to the editor used to create it.

The most trivial approach to whitelisting is to build a database of known-good, frequently used executable files – that is, EXE, COM, BAT and maybe a few kinds of scripts – and deny execution to any executable program not listed in the database. Sadly, this leaves the door open for many other ways in which a piece of malicious code could enter.

For instance, what about *Office* macros? Of course, once the producer of a whitelist-based protection program has thought about them, it is possible to make the program monitor macros in documents, too. But how many producers would think of them off-hand? And how many have the competence to handle obscure file formats – because you have to get to the macro bodies; you can’t just whitelist some *Word* documents and deny access to all other documents. And macros are just the beginning.

What about obscure scripting languages? Most producers would probably think of JavaScript and VBScript – but there are so many others, and most of them are sufficiently powerful to use to write viruses. For instance, what about the scripting languages of the various IRC clients (e.g. *mIRC*)? There are already many viruses written in these languages. Recently, a virus appeared that was written in the scripting language of the hex editor WinHex – how many producers of whitelist-based protections have heard of this scripting language? Other obscure scripting languages for which viruses exist include (but are not limited to): ABAP, ActionScript, Ami Pro macros, AutoLISP, Corel Draw! Script, Ferite, IDA script, KIX, Lua, MathLab script, One C, WinBatch, REG script, SQL, Perl, Python, Ruby… (Please don’t shoot me for calling the latter four ‘obscure scripting languages’.)

Furthermore, what about *Office* exploits? They arrive in a *Word*, *Excel* or *PowerPoint* document. Some obscure field in the document is corrupted, causing a buffer overflow somewhere in the *Office* application that opens it. This (the corruption) is the exploit. The exploit causes control to be transferred to a small piece of code that resides in the document too (usually, but not always, close to the corrupted field). This small piece of code is called ‘shellcode’. Then it usually extracts the real malicious program appended (often in encrypted form) after the end of the document – or downloads it from somewhere and runs it.

Now, a whitelist-based approach can prevent the dropped (or downloaded) executable from running. But it cannot stop the execution of the shellcode – not unless it stops the *Office* applications from running or disallows the opening of foreign documents – both of which would make the machine essentially unusable. And the shellcode doesn’t really *have* to drop an executable – it’s just easier to implement it this way. The shellcode runs directly in memory, in the context of the user who has opened the malicious document, and can do everything that the user is allowed to do. There is no hope for a whitelist-based approach of preventing that.

And what about threats that take control before the whitelist-based protection has had the chance to execute and then use stealth to disguise their presence? A boot sector virus would be the obvious example but there are many other convoluted ways in which malware can get itself executed during the computer’s startup process and before any whitelist-based protection.

As if all this wasn’t already bad enough – what about threats that do not exist as files at all? A typical example is the CodeRed virus [6]. It enters the attacked system as an HTTP GET request to port 80, exploiting a vulnerability in one of the system DLLs. It is never saved as a file (or any other object on the disk) – instead, it spreads memory-to-memory between the vulnerable machines on the internet.

How would a whitelist-based program protect against that? Even a blacklist-based approach (i.e. a scanner) has trouble with this kind of virus, because there are no obvious files to scan. But at least one can implement a packet scanner (again a blacklist-based approach) and block the requests sent by the virus before they reach the vulnerable DLL. Whitelisting network packets is a hopeless task.

Which programs are good?

The question of whether a program performs only legitimate actions is again an unsolvable one, in the general sense – just like, in general, it is impossible to answer the question of whether a program is a virus or whether it will stop after a finite number of steps (the so-called Halting Problem). The proof that these questions are unsolvable is a constructive one. That is, if someone claims to have invented an algorithm that can solve any one of these questions, the proof shows how to construct a program for which the algorithm will give incorrect results.

But, again, the above is just theory. In practice, there are other, more immediate problems. Let us suppose that you have decided to protect your computer from malware using a whitelist-based approach. For this, you need two things – a *program* that implements the approach and a *whitelist* (i.e. a database of legitimate programs that the program would allow to execute, while blocking everything else). OK, the program you purchase from some software producer – but what about the whitelist?

There are three possible solutions: you can purchase such a whitelist from somewhere; you can build one yourself; or you can use a combination of the first two approaches. Unfortunately, each of these approaches has its own problems.

Scalability, or the problems of global whitelists

Let us suppose that you decide to obtain your whitelist from a third party. This could be the same producer who sold you the whitelist-based program – or it could be an independent vendor. In both cases, however, you are most certainly not the only customer this vendor has, and the other customers most certainly don't use exactly the same 'good' programs that you do.

So, in order to serve all its customers, the whitelist vendor will have to compile some kind of global whitelist – a list of all legitimate programs in existence – in order to make sure that all programs their customers are likely to have are covered. In fact, there are already several companies trying to do just that – *Securewave*, *Bit9*, *AppSense*, etc. Unfortunately, as it turns out, their task is much, much more difficult than the task of the anti-virus companies – and for the same reasons, too.

At the recent International Antivirus Testing Workshop in Reykjavik, Iceland, I attended a presentation by Mario Vuksan from *Bit9* [7]. According to him, the company is trying to build just such a global whitelist – and is having a very hard time doing it. As it turns out, there are many more legitimate programs than malicious programs out there – many orders of magnitude more, as a matter of fact. Worse, the rate of creation of legitimate programs far exceeds the rate of creation of malicious ones – and it keeps increasing.

According to Mr Vuksan, just *Microsoft*, *IBM*, *SourceForge* and *Mozilla.Org* produce, respectively, 500 K, 100 K, 500 K and 250 K new executables every day! Currently, *Bit9* has 2.7 billion files listed in its global whitelist, aiming for 10 billion (that's US billion) – and is nowhere near finished. Just the index of the database is more than a hundred gigabytes. So, the joke about the malicious programs soon outnumbering the legitimate ones is just a joke – there is no chance of this happening any time soon (or even at all).

Clearly, the glut problem faced by a global whitelist producer is much worse than the problem faced by the average scanner producer. After all, we have to deal with 'only' about 5,000+ malicious programs per month.

This leads to other problems as well. First, how is the company going to deliver this database (the global whitelist) to its customers' computers? Delivering the whole of it is clearly out of the question – who will be willing to dedicate more than a hundred gigabytes to the database used by their malware protection? Not to mention that it will have to be updated with a couple of million new entries every day – much worse than the oh-so-hated regular scanner updates. (And the producer is unlikely to be willing to update it for free, either.)

Keeping it on the servers of the database producer and having it accessed remotely is not good, either. What is the producer going to tell its customers – 'sorry, you can't use your computers today, because we (or you) have a network failure and we can't check right now whether the program you want to run is legitimate or not'?

And, can you honestly believe that a company examining a couple of million new executables per day is not going to make mistakes and put malicious ones on the list? Even the anti-virus people tend to make an occasional mistake when telling the good programs from the bad – and these people are experts and their workload is orders of magnitude lower!

In addition to the technical problems, a global whitelist raises a political one – pretty much like centralized code signing. It is potentially dangerous to allow one entity to control the process. On the one hand, how can you be sure that a small producer's program will be included? And, on the other hand, how can you be sure that a big company like *Sony BMG* will not pay the whitelist producer to include its latest rootkit as a legitimate program in their database?

User (in)competence, or the problems of local whitelists

The alternative to using a whitelist somebody has built for you is to use one you have built yourself. Then you do not have the problem of millions of new legitimate programs per day – because the software installed on your computer is, more or less, constant. Sadly, this approach has problems of its own.

The main problem is that it relies on the user being competent enough to build such a whitelist. It is much more difficult than just running some sort of program that would scan your disks for executable programs and build a database of checksums for them.

For instance, how do you know that your system is not already infected? For most users, the only way to know this is by running a virus scanner. Oops, there goes the hope that conventional anti-virus is 'dead'. In addition, many users resort to seeking some kind of protection from malware only *after* their machines become infected.

Furthermore, what are you going to do when you want to install a new program? Obviously, you will have to update the whitelist in order to record the new program as legitimate and allow it to be executed. Of course, in order to do that, you will have to be able to decide whether the new program is legitimate. But if users in general were able to do that, they wouldn't get infected in the first place and there wouldn't be any need for anti-virus software (no matter whether whitelist- or blacklist-based) – because everybody would be installing only legitimate programs on their computers!

This approach is applicable only to tightly controlled corporate environments, where security is paramount (and takes precedence over usability), where the local whitelist is built by a competent security administrator, contains a relatively small number of programs, and the users are strictly forbidden from installing and running anything new. Sadly, it is totally unusable in home-user (and even in most corporate) environments.

The sum is less than its parts, or the problems of the combined approach

If neither of the above two approaches really works, then how about a combination of them? For instance, the whitelist-based protection vendor could supply you with a relatively small database of 'most popular legitimate programs' and also give you the possibility to update the database locally with any programs that you use but which are not included in it. Maybe if you do that, the combination of the two approaches will cancel each other's deficiencies?

Unfortunately, this is not the case. The combined approach just combines the problems of the above two approaches. For instance, who can decide what are 'the most popular programs'? And they are not a static set, either – so the database will have to keep increasing (or at least keep changing) – meaning that the product will have to be constantly updated – just like a conventional scanner.

And if you are allowed to put your programs on the whitelist, you are most likely to 'whitelist' some malicious program by mistake or due to lack of competence.

WHY PEOPLE MOSTLY USE SCANNERS

Malicious programs have been with us for more than a quarter of a century. Yet people still mostly use known-malware scanners to protect themselves from such programs. Now, I would be the first to admit that scanners are the weakest kind of protection from malware. Why, then, do people still rely on them almost exclusively?

It is not, as some people would have you believe, because of some kind of dark conspiracy among the anti-virus producers who want to keep getting money from you for their updates. It is because this is what the free market has established, no matter whether we like it or not.

The fact is that the average user is not interested in becoming a security expert. The users just want to be left alone doing their jobs, playing games, surfing the web. They start thinking about security only when some malware bites them.

A known-malware scanner is something the user can easily understand and use. It tells the user 'no, your computer is not infected' or 'yes, your computer is infected with the XYZ virus; do you want me to remove it?'. As opposed to that, the other kinds of anti-malware protection schemes require a significant level of competence from the user, in order to be understood and used correctly.

A heuristic analyser would say, 'The file Foo may contain a virus'. Well, does it, or doesn't it? A firewall would say, 'The program svchost.exe is communicating over port 1900'. What the heck does that mean and should it be permitted? A behaviour blocker would say, 'The program msvc.exe is trying to write to file Blah.exe'. Is that a virus attack? A whitelist-based protection would either occasionally deny the running of a program the user wants to run, or would ask the user whether the program is legitimate and should be added to the whitelist – a decision the user is generally not equipped to make.

This is why most users keep buying scanners – because they are easy to understand and easy to use. Yes, they are the weakest line of defence against malware – but this is what the users want to buy. And since we, the anti-virus producers, have to eat too, this is what we have been making and selling.

We would gladly sell the users something more secure and some of us have tried to do so over the past couple of decades. Does anybody remember Fred Cohen's *Integrity Shell*? The program *Untouchable*? *Integrity Master*? These were all generic, integrity-based products that were very efficient at stopping virus infections. Nevertheless, none of these products are manufactured any longer. The users voted with their wallets and the companies making these products either went out of business or switched to something else.

The same will happen to whitelisting, which is essentially a form of integrity checking. Any company that makes a product based exclusively on this approach will ultimately fail – because it will sell to only a relatively very small set of customers.

Some anti-virus companies will include whitelisting-based protection in their suites – and the users, in general, will happily keep using only the scanner part of these suites – the part they understand.

USE THE FORCE, LUKE

From what I have written so far about whitelisting, some readers could be left with the impression that it is a very bad idea, that it does not work and should be avoided like the plague. Nothing could be further from the truth.

As mentioned in the previous section, whitelisting is essentially a form of integrity checking – and I am a very strong proponent of integrity-based malware protection schemes, because malware (and especially computer viruses) are essentially an integrity problem.

With this article I am just trying to emphasize that whitelisting alone is incapable of stopping malware efficiently and is fraught with problems – just like blacklisting, only different kinds of problems. The proper way to protect from malware is by implementing defence in depth, not by relying on any particular single approach.

For example, use scanning to ensure that the system is initially malware-free. Use an integrity checker to ensure that it is not modified without authorization at a later date. Use behaviour blockers to detect intrusions. Use a personal firewall – not only to protect from external attacks but also as a kind of behaviour blocker – to detect if a program on your machine is trying to 'phone home' without your knowledge. Use sandboxing to isolate and misdirect potentially troublesome programs. Use encryption to hide the information that does not have to be visible all the time. Use backups!

Unfortunately, I understand very well that this is all just wishful thinking on my part. While a handful of geeks like me might be willing to understand what all of the above means and know how to install and operate it successfully on one's computer, it will remain forever way above the head of the average user who will happily continue using inefficient known-malware scanners.

CONCLUSION

Whitelisting, per se, is totally incapable of replacing conventional scanners as far as the general public is concerned. It does work well in small and tightly controlled environments, where security is more important than convenience, but the average home user (and most corporate users, as well) will never be able to rely on it exclusively.

It would be best if whitelisting were combined with blacklisting (and with other anti-virus techniques) to establish defence in depth. Sadly, most users do not have the competence required to build, understand, and maintain such defence – while they are able to use and understand scanners. This is why conventional scanning will be with us for a long, long time.

REFERENCES

[1] Robin Bloor. The decline of antivirus and the rise of whitelisting. http://www.theregister.co.uk/2007/06/27/whitelisting_v_antivirus/.

[2] Vesselin Bontchev. Methodology of Computer Anti-Virus Research. Ph.D. thesis. University of Hamburg, Germany, Chapter 4.

[3] Michael Venable, Andrew Walenstein, Matthew Hayes, Christopher Thomson and Arun Lakhotia. VILO: a shield in the malware variation battle. Virus Bulletin, July, 2007, pp.5–9.

[4] Fred Cohen. Automated Integrity Maintenance for Viral Defense. IFIP-TC11 Computers & Security, 1990.

[5] Kurt Wismer. The rise of whitelisting. Available from http://anti-virus-rants.blogspot.com/2006/03/rise-of-whitelisting.html.

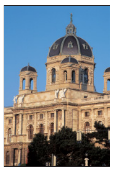
[6] Symantec Security Response. CodeRed Worm. Available from http://www.symantec.com/security_response/writeup.jsp?docid=2001-071911-5755-99&tabid=2.

[7] Mario Vuksan. Building and Leveraging White Database for Antivirus Testing. International Anti-Virus Testing Workshop, Reykjavik, Iceland, May 2007. Available from http://www.slideshare.net/frisksoftware/building-leveraging-white-database-for-antivirus-testing/download.

CALL FOR PAPERS

VB2007 VIENNA – CALL FOR LAST-MINUTE PAPERS

Virus Bulletin is seeking submissions from those wishing to present last-minute technical papers at VB2007, which will take place 19–21 September 2007 at the Hilton Vienna, Austria.



As usual, the conference will include a programme of 40-minute presentations running in two concurrent streams: Technical and Corporate, the running order for which has already been finalised and can be seen at http://www.virusbtn.com/conference/vb2007/programme/.

In addition to the traditional 40-minute presentations, a portion of the technical stream has been set aside for last-minute technical presentations.

Last-minute presentations will be selected by a committee consisting of members of the *VB* advisory board. The committee will be looking for presentations dealing with up-to-the-minute specialist topics.

There is no limit on the number of proposals that can be submitted/presented by any individual, and presenting a full paper does not preclude an individual from being selected to give a last-minute presentation.

Those selected for the last-minute presentations will be notified 7–10 days prior to the conference start, and will be required to prepare a 20-minute presentation (including time for questions) to be given on the afternoon of Thursday 20 September.

Those selected for the last-minute presentations will receive a 50% discount on the conference registration fee.

HOW TO SUBMIT A LAST-MINUTE PAPER PROPOSAL

Proposals must be sent to editor@virusbtn.com no later than **Monday 3 September 2007**. Submissions received after this date will not be considered. Please include full contact details with each submission.

FEATURE

PROFILING BINARIES FOR INSTRUMENTATION

Aleksander Czarnowski
AVET Information and Network Security, Poland

Instrumenting binaries is an analysis technique that is gaining popularity among security researchers for identifying vulnerabilities or analysing malware. However, before one starts on the instrumentation process it is worth profiling the target in order to better control further analysis. To demonstrate, let me use an example.

THE EXAMPLE

With *Visual Studio.NET 2005*, *Microsoft* introduced a buffer overflow protection scheme (enabled by the /GS switch) based on cookies. The idea behind this safeguard is very similar (to say the least) to the *Immunix Stackguard* solution. There are already a couple of good publications detailing how to bypass both /GS and *Stackguard*. We will use the code of the /GS safeguard as an example for further discussion.

RATIONALE

Quick identification of safeguards within binaries can be important during a security audit of binary objects. It can be helpful either during the instrumentation phase or during a static binary audit based strictly on code analysis.

When we know that a binary contains /GS buffer overflow protection we can:

- Write an exploit that will bypass /GS (we still need to remember about DEP too!).
- Write a non-executable stack exploit (employing a return-to-glibc style technique).
- Better model the threat profile for the application.

But wait a minute! Aren't *Windows 2003* and newer versions of *Windows* compiled with /GS enabled on all binaries? Yes they are, and we have to assume that every driver, library, service and application that comes with a system is /GS enabled. However, the same might not be true for third-party applications.

The real motivation for this work lies in the binaries compiled by a system integrator deployed in an audited system. To model threats and perform risk-based assessment we need to identify safeguards and evaluate their strength. Finding a third-party binary like a DLL or server application in a tested system certainly requires some kind of analysis. Even if you have access to the source code for those binary objects, not all safeguards or vulnerabilities can be identified at source level. While fixing defects, flaws and errors at the source code level is critical, evaluating the security of executed code cannot be skipped either.

TEST SET

There are several different tests that allow quick identification of the /GS safeguard within a binary:

1. Test if the binary has been compiled with *VS.NET 2005* or newer. If not, then it is not possible for the /GS safeguard to be present – however other similar safeguards could have been deployed (especially if the object is not in PE format but ELF – however this would apply to Unix binaries which are not discussed here).
2. Look for the security_init_cookie() function within the code – while the behaviour of the /GS option behaviour has been changed by *Microsoft* between releases of Visual Studio, the function is still present.
3. Analyse the import table for GetTickCount() and other functions to find whether the security cookie is being initialized.
4. Search prologs and epilogs of main functions for setting up cookies and checking return address integrity.

Test 1: has the binary been compiled with VS.NET 2005?

This method has two drawbacks: first, it needs to be upgraded every time a new VS compiler (or service pack) comes out. Secondly, detecting the type of compiler can be time consuming if automatic tools fail to identify the compiler or linker. Unless the binary is protected against disassembly this should not usually happen. Tools like *PeID* can easily identify the type of compiler. However, identifying the type of compiler still will not tell us whether /GS has been enabled. While it is enabled by default in *VS.NET 2005*, a programmer could disable it manually. So this method is not sufficient unless some additional tests are performed.

Test 2: is __security_init_cookie() present?

The /GS mechanism needs initialization which is performed inside the __security_init_cookie() function. This function takes no parameters and it is called near the original entry point. For example, the *Windows Console* application that uses _tmain() has the following entry point:

```
wmainCRTStartup proc near
        call    __security_init_cookie
        jmp     __tmainCRTStartup
wmainCRTStartup endp
```

So __security_init_cookie() is being called even before SEH has been set up (take a look at the _tmainCRTStartup code to confirm this).

The compile code of __security_init_cookie() is as follows:

```
.text:004018B0 __security_init_cookie proc near
; CODE XREF: wmainCRTStartup10p
.text:004018B0
.text:004018B0 PerformanceCount= LARGE_INTEGER ptr -
10h
.text:004018B0 SystemTimeAsFileTime= _FILETIME ptr -8
.text:004018B0
.text:004018B0      push    ebp
.text:004018B1      mov     ebp, esp
.text:004018B3      sub     esp, 10h
.text:004018B6      mov     eax, __security_cookie
.text:004018BB      and
[ebp+SystemTimeAsFileTime.dwLowDateTime], 0
.text:004018BF      and
[ebp+SystemTimeAsFileTime.dwHighDateTime], 0
.text:004018C3      push    ebx
.text:004018C4      push    edi
.text:004018C5      mov     edi, 0BB40E64Eh
.text:004018CA      cmp     eax, edi
.text:004018CC      mov     ebx, 0FFFF0000h
.text:004018D1      jz      short loc_4018E0
.text:004018D3      test    eax, ebx
.text:004018D5      jz      short loc_4018E0
.text:004018D7      not     eax
.text:004018D9      mov
__security_cookie_complement, eax
.text:004018DE      jmp     short loc_401940
.text:004018E0 ;
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
.text:004018E0
.text:004018E0 loc_4018E0:        ; CODE XREF:
__security_init_cookie+2110j
.text:004018E0      ; __security_init_cookie+2510j
.text:004018E0      push    esi
.text:004018E1      lea     eax,
[ebp+SystemTimeAsFileTime]
.text:004018E4      push    eax     ;
lpSystemTimeAsFileTime
.text:004018E5      call
ds:__imp__GetSystemTimeAsFileTime@4 ;
GetSystemTimeAsFileTime(x)
.text:004018EB      mov     esi,
[ebp+SystemTimeAsFileTime.dwHighDateTime]
.text:004018EE      xor     esi,
[ebp+SystemTimeAsFileTime.dwLowDateTime]
.text:004018F1      call
ds:__imp__GetCurrentProcessId@0 ;
GetCurrentProcessId()
```

```
.text:004018F7     xor    esi, eax
.text:004018F9     call
ds:__imp__GetCurrentThreadId@0 ; GetCurrentThreadId()
.text:004018FF     xor    esi, eax
.text:00401901     call    ds:__imp__GetTickCount@0 ;
GetTickCount()
.text:00401907     xor    esi, eax
.text:00401909     lea    eax, [ebp+PerformanceCount]
.text:0040190C     push   eax    ; lpPerformanceCount
.text:0040190D     call
ds:__imp__QueryPerformanceCounter@4 ;
QueryPerformanceCounter(x)
.text:00401913     mov    eax, dword ptr
[ebp+PerformanceCount+4]
.text:00401916     xor    eax, dword ptr
[ebp+PerformanceCount]
.text:00401919     xor    esi, eax
.text:0040191B     cmp    esi, edi
.text:0040191D     jnz    short loc_401926
.text:0040191F     mov    esi, 0BB40E64Fh
.text:00401924     jmp    short loc_401931
.text:00401926 ;
¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.text:00401926
.text:00401926 loc_401926:
; CODE XREF: __security_init_cookie+6D11j
.text:00401926     test   esi, ebx
.text:00401928     jnz    short loc_401931
.text:0040192A     mov    eax, esi
.text:0040192C     shl    eax, 10h
.text:0040192F     or     esi, eax
.text:00401931
.text:00401931 loc_401931:
; CODE XREF: __security_init_cookie+7411j
.text:00401931
; __security_init_cookie+7811j
.text:00401931     mov    __security_cookie, esi
.text:00401937     not    esi
.text:00401939     mov
__security_cookie_complement, esi
.text:0040193F     pop    esi
.text:00401940
.text:00401940 loc_401940:
; CODE XREF: __security_init_cookie+2E11j
.text:00401940     pop    edi
.text:00401941     pop    ebx
.text:00401942     leave
.text:00401943     retn
.text:00401943 __security_init_cookie endp
```

As you can see this function takes no parameters, so despite the fact that it is being called near the original entry point there is nothing special that could help us in its identification by using calling convention and/or passed parameters. In such a case we need to use the function code as a signature.

Take a look at this code snippet:

```
.text:004018C5     mov     edi, 0BB40E64Eh
.text:004018CA     cmp     eax, edi
.text:004018CC     mov     ebx, 0FFFF0000h
```

It seems that we can use the 0xBB4OE64E value as a signature. Three bytes further on there is another value we can use in a signature: 0xFFFF0000. If we find such a signature within the code section of a binary it is probable that it contains code for a /GS safeguard.

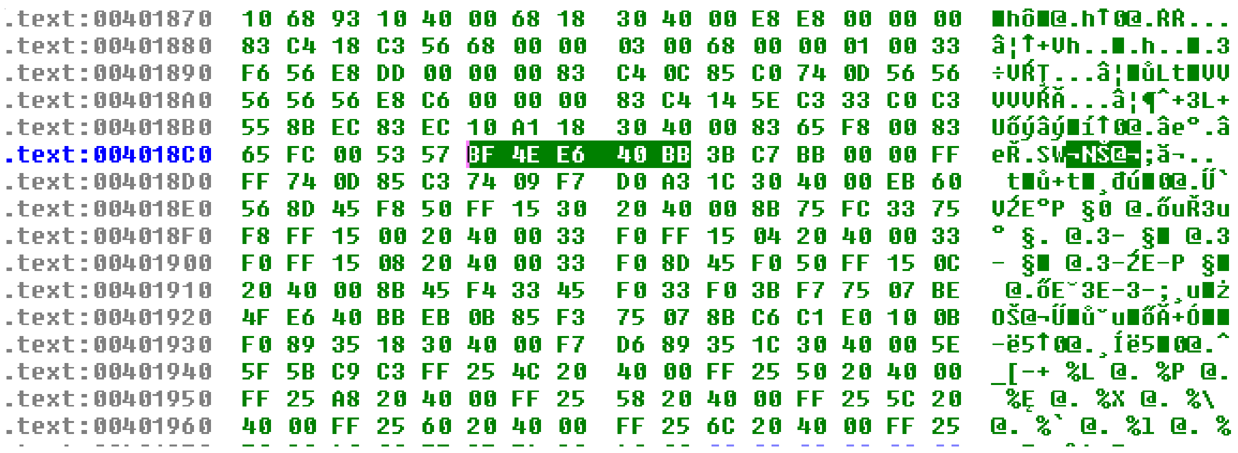
We can perform an additional test – take a look at address 0x004018B6:

```
.text:004018B6     mov     eax, __security_cookie
```

The value of __security_cookie is set to 0x0BB4E64E:

```
.data:00403018 __security_cookie dd 0BB40E64Eh
```

So we can look for 0x0BB4E64E twice: once in the code section (.text) and once in the data section (.data). If we find both occurrences in both sections we can assume that the binary has been compiled with /GS code. The whole code of



```
.text:00401870  10 68 93 10 40 00 68 18  30 40 00 E8 E8 00 00 00  ■hô■@.h↑0@.RR...
.text:00401880  83 C4 18 C3 56 68 00 00  03 00 68 00 00 01 00 33  â¦↑+Vh..■.h..■.3
.text:00401890  F6 56 E8 DD 00 00 00 83  C4 0C 85 C0 74 0D 56 56  ÷VŘŢ...â¦■ůLt■VV
.text:004018A0  56 56 56 E8 C6 00 00 00  83 C4 14 5E C3 33 C0 C3  VVVŘĂ...â¦¶^+3L+
.text:004018B0  55 8B EC 83 EC 10 A1 18  30 40 00 83 65 F8 00 83  Uőýâý■í↑0@.âe°.â
.text:004018C0  65 FC 00 53 57 BF 4E E6  40 BB 3B C7 BB 00 00 FF  eŘ.SW¬NŠ@¬;ă¬..
.text:004018D0  FF 74 0D 85 C3 74 09 F7  D0 A3 1C 30 40 00 EB 60   t■ů+t■¸đú■0@.Ű`
.text:004018E0  56 8D 45 F8 50 FF 15 30  20 40 00 8B 75 FC 33 75  VŹE°P §0 @.őuŘ3u
.text:004018F0  F8 FF 15 00 20 40 00 33  F0 FF 15 04 20 40 00 33  ° §. @.3- §■ @.3
.text:00401900  F0 FF 15 08 20 40 00 33  F0 8D 45 F0 50 FF 15 0C  - §■ @.3-ŹE-P §■
.text:00401910  20 40 00 8B 45 F4 33 45  F0 33 F0 3B F7 75 07 BE   @.őE˘3E-3-;¸u■ż
.text:00401920  4F E6 40 BB EB 0B 85 F3  75 07 8B C6 C1 E0 10 0B  OŠ@¬Ű■ů˘u■őĂ+Ó■■
.text:00401930  F0 89 35 18 30 40 00 F7  D6 89 35 1C 30 40 00 5E  -ë5↑0@.¸Íë5■0@.^
.text:00401940  5F 5B C9 C3 FF 25 4C 20  40 00 FF 25 50 20 40 00  _[-+ %L @. %P @.
.text:00401950  FF 25 A8 20 40 00 FF 25  58 20 40 00 FF 25 5C 20   %Ę @. %X @. %\
.text:00401960  40 00 FF 25 60 20 40 00  FF 25 6C 20 40 00 FF 25  @. %` @. %l @. %
```

Figure 1: Signature used within the __security_init_cookie() function.

the __security_init_cookie() function can also be used as a signature. This method can be extended further by looking for other /GS functions like:

- __security_check_cookie()
- __report_gs_failure()

Of course, searching a binary for signatures based on initial values will not work if *Microsoft* changes them in the next compiler.

Test 3: test for imported functions by __security_init_cookie()

If you have analysed the compile code of __security_init_cookie() you will already know that it uses the following functions:

- GetSystemTimeAsFileTime(x)
- GetCurrentProcessId()
- GetCurrentThreadId()
- GetTickCount()
- QueryPerformanceCounter()

This leads us to a simple conclusion: if we find such imports in the Import Address Table (IAT) then there is a chance that the binary contains /GS code. Parsing the IAT table (providing it is not protected and the binary is not compressed) is trivial, reliable and can easily be implemented inside the static binary audit process.

Test 4: prolog and epilog analysis

This method works very well in the case of dynamic analysis (or code emulation) and can easily be automated. If you are tracing (or 'stalking') code you just need to intercept the call instructions. After a call is executed the prolog function can be analysed. However, remember that not every function being called (either by call or by push/jmp) uses the /GS safeguard. Here is an example:

```
.text:00401000 Base__demo      proc near
; DATA XREF: .rdata:const Base::'vftable' o
.text:00401000     push   offset aBaseClass ; "Base
class\n"
.text:00401005     call   ds:__imp__wprintf
.text:0040100B     pop    ecx
.text:0040100C     retn
.text:0040100C Base__demo endp
```

As you can see, this function is even missing the typical push ebp/mov ebp, esp prolog! So look for /GS code only if the function has a standard prolog and local variables are allocated on the stack.

PRACTICE

Unless you are using dynamic analysis, a combination of test method 3 with test method 2 is the most practical as these don't require full disassembly of the object.

Method 4 and looking for the execution of __security_init_cookie() is reasonable during dynamic analysis and doesn't incur a huge performance hit – especially if you disable the check for /GS code after the first hit of __security_init_cookie() and __security_check_cookie(). Similar rules apply in other cases.

Profiling can also be used for quick identification of compressed binaries, entry point obfuscation or internal use of strong cryptography.

THE PROCESS

We can break up the process into the following steps assuming PE file format:

1. Is it a PE file? Check the signature.
2. Is it 32- or 64-bit? Assume proper values.
3. Is it a .NET managed file?
4. Is it a DLL?
5. Are loader flags standard?
6. Is base address standard? No: could it be some anti-debugging technique?
7. Is IAT correct? No: could it be some anti-debugging technique or a compressed binary?
8. What entries are in IAT?
9. Scan the binary for cryptographic algorithm initialization values.

FINAL WORDS

A lot of anti-debugging schemes deploy some of the above techniques to protect application code. While the example described in this article is trivial, profiling can provide a lot of important information about the target we wish to analyse (assuming that the binary is not compressed internally [with *UPX* or other similar tools], its code is neither obfuscated nor encrypted, and the IAT table is intact).

Most profiling techniques are easy to implement and time efficient. What's more important is that they can be built on top of a debugger (like *OllyDBG*) or disassembler (like *IDA Pro*, e.g. the FindCrypto plug-in http://hexblog.com/2006/01/findcrypt.html) to aid the analysis process.

COMPARATIVE REVIEW

WINDOWS VISTA X64 BUSINESS EDITION

John Hawes

After the enormous number of entrants for the last VB100 comparative review (see *VB*, June 2007, p.10), I was hoping for a quieter time this month. *Vista* is still pretty new, and the 64-bit version would, I hoped, pose enough difficulties to frighten off all but the most serious (or foolhardy) of vendors. The operating system promised no shocks for me, having gained some experience with its 32-bit sister in the early days of its release, but I was pretty sure that at least some of the products submitted would exhibit those quirks which seem just about compulsory on new platforms.

The range of products submitted offered few surprises. With 20 entries, the comparative proved a little more popular than I had expected, but there were no brand new faces this time, with most of the field made up by the group of familiar names that rarely miss a VB100.

PLATFORM AND TEST SETS

64-bit *Vista* is, on the surface, no different from the 32-bit version used in the February tests (see *VB*, February 2007, p.14), and as identical hardware was used the experience of building the test systems for this comparative had more than the usual number of déjà vu moments. Under the bonnet the differences should be fairly minimal, with compatibility generally not supposed to be an issue, although much debate raged in the months prior to the platform's release over access to the 'PatchGuard' kernel protection system and other additional security measures added to *Vista* on 64-bit architectures. Added to the User Access Controls, which caused a few wobbles in the earlier test, these new items could be expected to upset at least some functionality, and I could only hope nothing would seriously impede the process of ploughing through all the tests.

Installing was a pleasantly speedy process, accompanied by the flashy visual gimmicks that typify the platform, and previous experience once again helped steer a course around the small changes that hide most of the system configuration tools. After the eye-straining experience of the earlier *Vista* test, I reverted to Luddite principles and set all the display options to '*Windows* Classic' styles, eschewing the luminous and the curvy in favour of familiar, boxy grey windows and menus. Otherwise no changes were made from the default setup other than configuring networking.

The April 2007 WildList was used for this test, which added fairly few new items to the current mix – a scattering of the regular names, W32/Bagle, W32/Netsky and so on, plus some new variants on the same theme and a reappearance of a real old timer, W32/Sober. A pretty large swathe also fell off the list this month, including several varieties of the W32/Looked infectors which only joined the list in the last few months; considerable numbers of the W32/Mytob, W32/Rbot, W32/Stration and W32/Sdbot variants which make up the bulk of the WildList also fell to one side.

Other test sets were added to in a small way, mostly by the expansion of polymorphic sets, but the biggest changes were made in the clean and speed sets, with a large swathe of items added. The additions mainly comprised popular home-user software gathered from the web, but also a sizeable set of business and development tools and products, from *Microsoft* among others. These added a large number of installers and packages to the archive set, and the expanded contents to the various other sets as appropriate.

One item from amongst the stash turned out to be a 'legitimate' keylogger tool which was, of course, deemed inappropriate for the speed tests. Having backed up the sets ready for testing, the first run revealed that the installer had failed to be removed from the false positive set, so it remained throughout the tests and became an interesting indicator of which products were covering this kind of unpleasantware.

Alwil avast! Professional 4.7.1015

ItW	100.00%	**Worms & bots**	99.77%
ItW (o/a)	100.00%	**DOS**	99.34%
File infector	98.39%	**Macro**	99.56%
Polymorphic	85.94%	**False positives**	0

Alwil's product started things off in the manner I expected things would carry on – with one of *Vista*'s endless queries about whether I really wanted to install this software from an unknown publisher. These queries are, of course, a security measure, but it is hard to avoid the conclusion that most users, bombarded with these popups, blocks and queries, will soon tire of them, cease to read the scant details provided and click 'OK' without further thought. Critics of the 'warning – are you sure?' method have argued that these systems do little more than indemnify *Microsoft* from accusations of failing to secure its operating system, passing all blame onto the foolish end-user, while I have often felt the sneaking suspicion that they have been put there merely to irritate people testing large numbers of software products.

Once the smooth and speedy install was done and the system rebooted, yet another popup demanded to know if I really meant to open the *avast!* interface, then I finally got to play around with it. Skipping straight past the stylized

On-access tests	ItW		Worms & bots		DOS		File infector		Macro		Polymorphic		Clean set	
	No. missed	%	No. missed	%	No. missed	%	No. missed	%	No. missed	%	No. missed	%	False positives	Susp.
Alwil avast!	0	100.00%	1	99.77%	236	99.34%	14	97.97%	18	99.56%	268	85.94%		2
Bullguard	0	100.00%	0	100.00%	11	99.44%	3	98.32%	22	99.46%	10	97.91%		
CA eTrust	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A		
CAT Quick Heal	0	100.00%	0	100.00%	1103	91.39%	23	96.16%	82	98.04%	388	76.99%		
ESET Nod32	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Fortinet FortiClient	1	99.97%	1	99.97%	0	100.00%	2	99.52%	821	81.05%	56	94.99%	1	
G DATA AntiVirusKit	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		2
Grisoft AVG	0	100.00%	3	99.59%	197	99.10%	16	97.10%	3	99.93%	194	76.46%		
Ikarus Virus Utilities	0	100.00%	1	99.92%	2119	92.93%	42	93.92%	174	95.94%	399	71.81%	46	
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	2	99.52%	0	100.00%	0	100.00%		
Kingsoft Internet Security	0	100.00%	429	14.33%	12937	55.59%	192	70.13%	463	89.92%	2202	31.90%		
McAfee VirusScan	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Microsoft Forefront	0	100.00%	0	100.00%	0	100.00%	5	98.62%	0	100.00%	29	96.30%		
Microworld eScan	0	100.00%	0	100.00%	0	100.00%	1	99.28%	15	99.69%	0	100.00%		
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	8	99.80%	0	100.00%		
Symantec AntiVirus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Trend Micro Client Server Security	0	100.00%	0	100.00%	744	98.39%	15	97.80%	13	99.68%	152	93.29%	1	
Trend Micro OfficeScan	0	100.00%	0	100.00%	744	98.39%	15	97.80%	13	99.68%	152	93.29%	1	
Trend Micro PC-cillin	0	100.00%	0	100.00%	744	98.39%	17	97.32%	13	99.68%	152	93.29%	1	
VirusBuster VirusBuster	0	100.00%	1	99.97%	20	99.77%	11	98.08%	0	100.00%	98	88.22%	1	4

basic version of the GUI, which I imagine may be quite simple to use for those practised in its intricacies but remains almost entirely baffling to me, I delved into the advanced version for most of my testing needs. From here settings can be changed by adjusting the properties of various 'tasks', and some tweaks to the settings of the 'resident protection' (on-access) and 'interactive scan' jobs proved adequate for most of the tests.

During on-demand scanning, the window area showing the status and results of the scan was a little wobbly, starting out completely blank and remaining so until some judicious jiggling of the scroll bars brought the information out of hiding. This allowed me to track the progress of scans and gather results, which showed pretty solid detection across all sets and decent speeds in the default settings, which do not include delving into compressed archives. With archive scanning enabled, things slowed to a bit of a crawl, particularly with a couple of .jar files which eventually had to be removed from the set to allow the scans to complete in reasonable time.

This aside, detection in the WildList proved faultless, and with just a joke program and a risky tool spotted in the clean sets, *avast!* easily picks up another VB100 award.

Bullguard v.7.0 x64

ItW	100.00%	**Worms & bots**	99.77%
ItW (o/a)	100.00%	**DOS**	99.77%
File infector	98.32%	**Macro**	99.69%
Polymorphic	97.94%	**False positives**	0

Plucky *Bullguard* stormed to victory in the June *XP* comparative (see *VB*, June 2007, p.10), seizing a well deserved VB100 award at first attempt. Returning for a second time, the product seemed much the same – slick and smoothly laid out and adorned with numerous wrinkly-faced pooches. The installation procedure requests a web-based login process to 'activate' the product, but this can be skipped to give a seven-day 'grace period'. The main product interface is a pretty affair, glossy and colourful and featuring some pleasantly quirky, friendly comments scattered amongst the more serious business of malware protection.

Operation was fairly straightforward, with configuration options not enormously granular but with most things required by the average home user amply covered.

Right-click scanning was available, but only functions in fully activated products, so I was reduced to using the interface itself for the on-demand tests – no great disaster really as the scanning section is as clearly designed as the rest of the GUI. Speeds were fairly good, considering the depth of scanning going on, and that rogue keylogger that crept into the clean set was spotted, and identified as containing both spying and hiding techniques.

An initial submission of the product proved to be a faulty build, missing a vital component which rendered the on-access scanner inactive after scanning 255 files. However, a fully working replacement suffered no such problems, and a suspected false positive in one of the clean sets, a file labelled as a spyware-doctored hosts file, proved to be a database of such subverted hosts files used by a security product, and was thus stricken from the test set. With no other problems, *Bullguard* earns itself a second VB100 award.

CA eTrust r.8.1.634.0

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	N/A	**DOS**	99.67%
File infector	99.38%	**Macro**	99.82%
Polymorphic	99.85%	**False positives**	0

CA's *eTrust* has a long and solid history in VB100 comparative testing. *CA*'s traditional submission method has been to provide a CD, or CD image, each time a major update to the main product is released, and in between simply to send in definition updates for each review. For this test, however, the submission method proved not to be good enough, with the 8.1 build which had been sitting cosily in the *VB* lab since the new year, proving inadequate for the demands of 64-bit *Vista*. The installer began its business happily, let me go through the lengthy process of scrolling through several sizeable EULAs and filling in lots of required user information, then quietly freaked out and froze. After some frantic pestering a more suitable version was eventually provided by the vendor, just in time to make the cut for this comparative.

The 64-bit version proved more effective, and after yet another run through the arduous install process I was able to get my hands on the product itself. The browser-borne GUI, usually a slow and unwieldy thing, was considerably more responsive than usual under 32-bit *Vista* earlier in the year, but any hopes of a repeat performance were soon dashed, and several long sessions of staring at the progress bar seemed to augur badly for the rest of the test. Fortunately, I discovered the right-click scanning option opened a mini-interface of its own, which was nice and simple and responsive, and carried enough configurability to run through the on-demand tests with ease. Detection was in the upper range as expected, and speeds were impressively zippy.

Moving on to the on-access side of things, speeds were even more remarkable. Suspiciously so, in fact. Trying the on-access detection test revealed something was seriously wrong – nothing seemed to be detected at all. I tried numerous methods beyond the simple opener tool which usually suffices to exercise *CA*'s products, but copying files around the system, and even dropping them in from the network, sparked neither blocking nor alerting. Several reinstallations on fresh systems failed to make things any better, and I was on the verge of despair when I discovered the root cause.

During on-demand speed testing, I had observed that checking the 'scan archives' box on its own had no effect, as the list of archive types remained unchecked – once all of these were selected, archives were indeed scanned internally. Changing some settings in the on-access controls, which sadly meant resorting to the full ITM interface, I found scanning suddenly worked fine; with 'scan all files' active, normal scores were recorded in all infected sets. It emerged that the default setting, targeting only a pre-defined list of extensions, was failing to work because the pre-defined list was entirely empty.

This being the default setting, testing could not successfully be carried out under the rules of the test, but hopefully most administrators would spot this flaw before deploying the software to their 64-bit *Vista* users. Nevertheless, it is enough of a problem to deny *CA* a VB100 award this time, and to keep its spectacular speed settings from cluttering our speed graphs.

CAT Quick Heal 2007 v.9.00

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	95.06%
File infector	97.00%	**Macro**	98.18%
Polymorphic	76.99%	**False positives**	0

CAT's *QuickHeal* installs as swiftly as its title implies, with nothing to tax the mind along the way, and the clear and well laid out interface is equally speedy to navigate, responsive and stable throughout the tests. The welcoming purple blob planted in the system carries a pleasant message congratulating the user on their choice of security software, and the whole product is set out in a similarly user-friendly manner.

A few oddities were encountered during testing: logs seemed to take a long time to export to file and the switch

On-demand tests	ItW		Worms & bots		DOS		File infector		Macro		Polymorphic		Clean set	
	No. missed	%	No. missed	%	No. missed	%	No. missed	%	No. missed	%	No. missed	%	False positives	Susp.
Alwil avast!	0	100.00%	1	99.77%	236	99.34%	12	98.39%	18	99.56%	268	85.94%		2
Bullguard	0	100.00%	1	99.77%	15	99.77%	3	98.32%	13	99.69%	5	97.94%		
CA eTrust	0	100.00%	0	100.00%	235	99.67%	3	99.38%	12	99.82%	2	99.85%		
CAT Quick Heal	0	100.00%	0	100.00%	1054	95.06%	20	97.00%	73	98.18%	388	76.99%		
ESET Nod32	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Fortinet FortiClient	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	1	
G DATA AntiVirusKit	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		4
Grisoft AVG	0	100.00%	2	99.69%	197	99.10%	14	97.58%	0	100.00%	194	76.46%		
Ikarus Virus Utilities	0	100.00%	1	99.92%	2119	92.93%	42	93.92%	158	96.27%	399	71.81%	46	
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Kingsoft Internet Security	0	100.00%	429	14.33%	12937	55.59%	192	70.13%	463	89.92%	2202	31.90%		
McAfee VirusScan	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Microsoft Forefront	0	100.00%	0	100.00%	0	100.00%	3	99.10%	0	100.00%	29	96.30%		
Microworld eScan	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	8	99.80%	0	100.00%		
Symantec AntiVirus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Trend Micro Client Server Security	0	100.00%	0	100.00%	233	99.47%	9	99.24%	13	99.68%	152	93.29%	1	
Trend Micro OfficeScan	0	100.00%	0	100.00%	744	98.39%	15	97.80%	13	99.68%	152	93.29%	1	
Trend Micro PC-cillin	0	100.00%	0	100.00%	233	99.47%	9	99.24%	13	99.68%	152	93.29%	1	
VirusBuster VirusBuster	0	100.00%	0	100.00%	20	99.77%	8	99.28%	0	100.00%	98	88.22%	1	4

from the main interface to the configuration area brought about one of those flashes of blackness which seem a regular occurrence under *Vista*, but other than these there was nothing to detract from the overall pleasant experience of using the product.

Scanning speeds were as decent as expected, although the option to scan inside archives and otherwise expand the scope of the on-access mode was notably absent, and while detection over the older sets remains less than flawless nothing was missed in the newer areas, including the WildList. With no false positives generated in the clean set, *QuickHeal* earns itself another VB100 award.

ESET Nod32 Antivirus System v.2375

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

ESET's upcoming overhaul of its product has yet to reach the *VB* test bench, so once again the tests were run with the familiar interface which has graced every *Windows* test of my reign here. The product's current design loses a lot of its glamour in the more glossy environment of *Vista*, but practice has nullified its oddities, which are mostly confined to identifying its component modules by inscrutable acronyms, and the interface has become a pleasure to use.

Tweaking the settings of 'AMON' and the right-click scan 'profile' to my needs, all the tests were carried out quickly and easily, testament to the solidity and lightning scanning speed of the engine powering the product as much as to the usability of the interface.

Speeds were a little less eye-opening than usual over the much expanded archive set. On-access settings cannot be expanded to cover the full range of files scanned on demand, and the product threw up some errors scanning the master boot records of my hard drives, but beyond these minor quibbles detection was as unimpeachable as ever. With nothing missed in any set and not a shadow of a false positive, *ESET* earns yet another VB100 award to add to its sizeable stash.

Fortinet FortiClient 3.0.458

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	99.97%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	1

FortiClient is another product that has changed little since I first encountered it, on the surface at least. Its busy interface covers a wide range of functionality, arranged into a long row of tabs squeezed down the left-hand side of the window and each further divided into more tabs for configuring and checking the status of each area. This wide range of functions caused even more questioning from *Vista*, with numerous confirmations required to install the various drivers etc. required by the product.

On-demand results were as comprehensive as ever, and scanning speeds were fairly decent, with particular thoroughness shown to the executable set, where a single item, part of a PDF creation utility, was flagged as vaguely ‘suspicious’. Under the tightened rules of the VB100 such a slander on a file’s reputation is adjudged enough to disqualify a product from the award.

This would have seemed rather a cruel treatment of a solid product had not a change to the default on-access settings, from ‘all files’ to only ‘programs and documents’, meant that besides large numbers of macro and polymorphic samples being missed thanks to the omission of .xls and .xlt files from the document set, a single WildList sample, W32/Funlove in .ocx format, was also passed over. Although detection for all these items was clearly in place, the VB100 rules insist on using default settings at all times, and it appears that in an attempt to improve its on-access performance, *Fortinet* may have reduced its coverage a little too far.

G DATA AntiVirusKit 17.0.7171

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

G DATA’s *AVK* has a very slick appearance, and equally smooth and impressive detection powers. The interface is clearly laid out, allowing all the required configuration for my needs without appearing too complex or technical for the average user. It has performed excellently in the last few tests with barely a slip or stumble to report.

This time, after installation and a reboot, things were as solid and stable as ever, with no surprises or annoyances beyond the rather odd habit of opening new instances of the interface each time the handy context-menu scan is used, leaving several strewn about the screen if a forgetful tester omits to shut them down. Logging was slightly pesky, with file names separated from their paths, but for those real-world people not needing to extract large amounts of data this is probably an extra touch of clarity and thoughtful design.

For a multiple-engine product, speeds were pretty decent in most sets, though the archive collection did take some serious time to slog through – the product helpfully warns about potential slowness if no maximum depth of archive scanning is set. The keylogger that slipped into the clean set was spotted – described as a ‘monitor’, ‘Not-a-virus’ – as well as a joke program, an *IRC* client and some ‘Risktools’, but nothing marred the superb detection and *G DATA* earns another VB100 award.

Grisoft AVG Professional Edition 7.5.476

ItW	100.00%	**Worms & bots**	99.69%
ItW (o/a)	100.00%	**DOS**	99.10%
File infector	97.58%	**Macro**	100.00%
Polymorphic	76.46%	**False positives**	0

Grisoft’s *AVG*, wildly popular with the home-user market thanks to the broad availability of its free version, remains a solid performer, and its installation was another simple and painless experience. The user interface itself is divided into simple and advanced versions, and while doubtless more than adequate for the needs of most, has always proved a little confusing when more in-depth configuration is required to smooth the passage of a test, but familiarity with its rather esoteric layout has improved matters considerably.

Speeds were on the slow side, but on-access overheads were considerably better than the more thorough on-demand settings; detection rates were similarly solid, if not flawless, and without a miss in the WildList set or a false positive to mark it down, *Grisoft* also makes the grade for the VB100.

Ikarus Virus Utilities 1.0.57

ItW	100.00%	**Worms & bots**	99.92%
ItW (o/a)	100.00%	**DOS**	92.93%
File infector	93.92%	**Macro**	96.27%
Polymorphic	71.81%	**False positives**	46

Ikarus returned to the *VB* test bench in the June *XP* comparative after a nearly six-year absence, with a new

On-demand throughput	Archive files - default		Archive files - all files		Binaries and system files - default		Binaries and system files - all files		Media & documents - default		Media & documents - all files		Other file types - default		Other file types - default	
	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)	Time (s)	Throughput (MB/s)
Alwil avast!	136	15.05	610	3.36	211	12.02	245	10.35	109	13.81	132	11.40	57	11.85	67	10.08
Bullguard	2386	0.86	2386	0.86	321	7.90	321	7.90	80	18.81	80	18.81	103	6.56	103	6.56
CA eTrust	14	146.22	706	2.90	55	46.10	61	41.57	23	65.43	24	62.70	20	33.77	22	30.70
CAT Quick Heal	497	4.12	823	2.49	66	38.42	67	37.85	49	30.71	49	30.71	38	17.77	39	17.32
ESET Nod32	726	2.82	726	2.82	270	9.39	270	9.39	32	47.03	32	47.03	25	27.02	25	27.02
Fortinet FortiClient	342	5.99	342	5.99	565	4.49	565	4.49	38	39.60	38	39.60	34	19.87	34	19.87
G DATA AntiVirusKit	1987	1.03	3402	0.60	337	7.52	338	7.50	94	16.01	119	12.65	78	8.66	133	5.08
Grisoft AVG	2246	0.91	2246	0.91	315	8.04	315	8.04	133	11.33	133	11.33	16	41.45	16	41.45
Ikarus Virus Utilities	124	16.51	339	6.04	200	12.68	208	12.19	39	38.59	41	36.71	70	9.65	70	9.65
Kaspersky Anti-Virus	74	27.66	1299	1.58	54	46.96	182	13.93	65	23.15	71	21.20	48	14.07	59	11.45
Kingsoft Internet Security	679	3.01	679	3.01	257	9.87	257	9.87	69	21.81	69	21.81	78	8.66	78	8.66
McAfee VirusScan	628	3.26	628	3.26	322	7.88	322	7.88	46	32.72	46	32.72	53	12.74	53	12.74
Microsoft Forefront	550	3.72	550	3.72	195	13.00	195	13.00	54	27.87	54	27.87	32	21.11	32	21.11
Microworld eScan	1401	1.46	1401	1.46	460	5.51	460	5.51	276	5.45	276	5.45	280	2.41	280	2.41
Sophos Anti-Virus	23	89.00	713	2.87	217	11.69	234	10.84	37	40.67	54	27.87	29	23.29	65	10.39
Symantec AntiVirus	450	4.55	450	4.55	169	15.00	169	15.00	53	28.39	53	28.39	44	15.35	44	15.35
Trend Micro Client Server Security	74	27.66	79	25.91	167	15.18	169	15.00	21	71.66	22	68.41	30	22.51	32	21.11
Trend Micro OfficeScan	96	21.32	209	9.79	201	12.62	218	11.63	38	39.60	38	39.60	35	19.30	43	15.71
Trend Micro PC-cillin	203	10.08	212	9.66	180	14.09	183	13.86	26	57.88	26	57.88	30	22.51	34	19.87
VirusBuster VirusBuster	262	7.81	607	3.37	322	7.88	323	7.85	26	57.88	54	27.87	16	42.22	39	17.32

product in the later stages of development. On that occasion the product showed signs of needing a little more work. The submission method was a little different this time, with a CD image provided rather than the bare bones of the product itself. This smoothed over a few of the wrinkles previously experienced in the installation process – the requirement for the *.NET* framework, in this case installed automatically as part of the setup process, being the most obvious.

Running of the product itself, after an apparently unstoppable attempt to contact the web to update, had also improved considerably. Double-clicking the desktop icon at first had no effect, leading to fears of a repeat of earlier problems with starting the GUI, but it proved just to be rather slow to open. Some of the language used is rather odd, and occasionally seems misleading – the on-access monitor reports it is 'inactive' if automatic updating is not running, which may actually be a useful warning to users that running out-of-date software is a dangerous state to be in.

There were a few further problems with the responsiveness of the interface during the more stressful of the detection tests; things faded out while scanning the infected collections and again after I foolishly clicked the 'quarantine' button with several thousand files waiting to be dealt with. The results showed that, while detection across the zoo collections was a little lacking, the WildList set was fully covered in both modes.

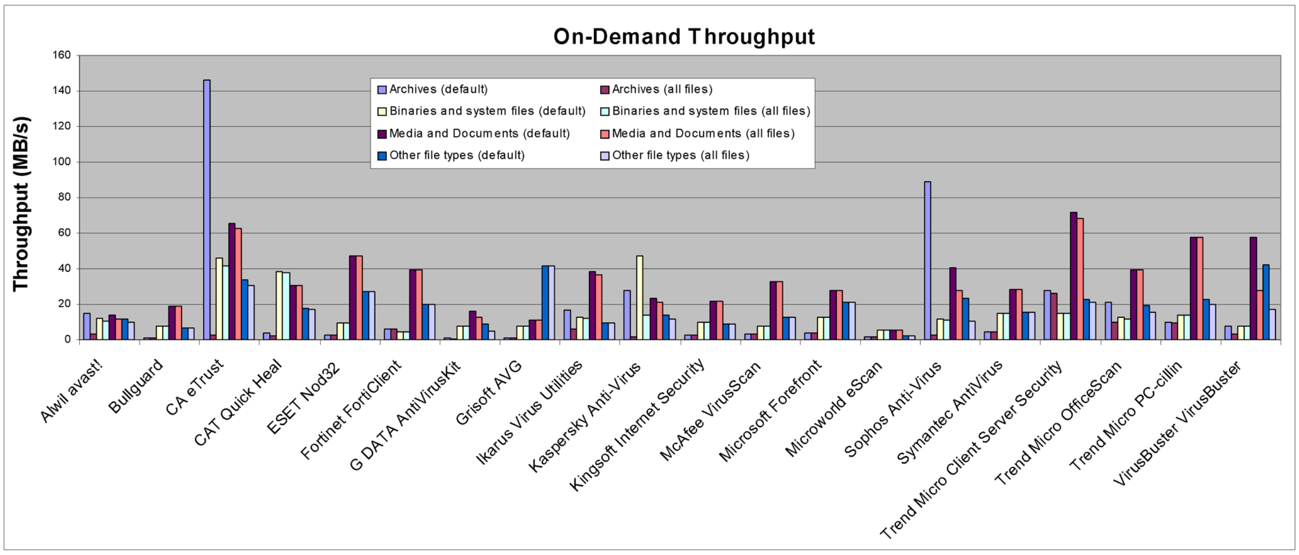
Hopes that *Ikarus* may have qualified for its first VB100 were dashed in the clean sets however; speeds were perfectly reasonable throughout, and outstanding in some sets, but the scans were marred by a scattering of false positives across several of the sets. The much improved product will surely make the grade sometime soon.

Kaspersky Anti-Virus 7.0.0.123

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

Version 6 of *Kaspersky*'s product has put in some sterling performances in VB100 testing over the last year or so (a momentary lapse which denied it the award in the last test notwithstanding), and the product impressed me considerably in a more thorough standalone review some months ago. Now it has been superseded by a new edition, with some serious redesign work having been put into the appearance of the product.

The install process looked somewhat shinier, but also felt a little slow moving, and required a reboot to complete. The new interface was considerably more glossy than the



previous incarnation, and has lost the cuddly, cartoon-like appearance in favour of a more high-tech, space-age theme.

The redesign has not reduced the fine-grained configurability of the product, or the solid thoroughness of the detection – a thoroughness which is reflected by the speed measurements, particularly with archive scanning enabled. No false positives and immaculate detection levels, barring a couple of files in formats not scanned by default on access, brings *Kaspersky* back to the podium as a VB100 winner.

Kingsoft Internet Security 2007.6.21.206

ItW	100.00%	**Worms & bots**	14.33%
ItW (o/a)	100.00%	**DOS**	55.59%
File infector	70.13%	**Macro**	89.92%
Polymorphic	31.90%	**False positives**	0

Kingsoft makes its second attempt at the VB100 this month, having first entered several months ago (see *VB*, October 2006, p.10). The product's earlier appearance was marred by a small number of misses in the WildList test set and a fair number of false positives, but the company has reportedly been working hard to resolve these issues in preparation for its latest submission.

Running through the installer and the process of navigating around the product proved a happy experience, with everything running smoothly and slickly with a minimum of pestering from *Vista*. A problem encountered previously, with the log display interface lacking translation and crashing out when a log was selected, proved avoidable by the simple expedient of switching the system locale from the UK version usually used in *VB* tests to the more standard US setting.

Speeds in the clean sets were good, and no false positives were flagged in any of the newly enlarged sets. In the infected sets, detection rates were low and in some cases very low – most worryingly in the worms and bots set which contains the newest material, much of it recently downgraded from the WildList. *Kingsoft*'s developers have clearly been focusing closely on the WildList itself, however, and much to their credit the product managed to cover the whole set, earning it a VB100 award at its second attempt.

McAfee VirusScan Enterprise v.8.5l

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

McAfee's corporate product remains its familiar self, somewhat severe and serious in appearance and reliable in performance. The installer was slick and problem-free, with no reboot required, but to open the interface required yet another confirmation dialog every time, which proved a little annoying.

A previous annoyance, that the control to disable and activate the on-access scanner could not be run from the interface but required using the system tray menu, has been resolved in this version, speeding the tests along nicely, and with decent speeds and flawless detection, *McAfee* wins itself another VB100 award.

File access lag time	Archive files - default		Archive files - all files		Binaries and system files - default		Binaries and system files - all files		Media & documents - default		Media & documents - all files		Other file types - default		Other file types - default	
	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)
Alwil avast!	49	0.02	1410	0.69	223	0.08	252	0.09	120	0.07	133	0.08	59	0.07	58	0.06
Bullguard	111	0.05	1152	0.56	297	0.11	341	0.13	82	0.04	96	0.05	109	0.14	119	0.16
CA eTrust	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CAT Quick Heal	20	0.01	N/A	N/A	67	0.02	N/A	N/A	26	0.01	N/A	N/A	18	0.01	N/A	N/A
ESET Nod32	8	0.00	N/A	N/A	60	0.02	N/A	N/A	47	0.02	N/A	N/A	36	0.03	N/A	N/A
Fortinet FortiClient	89	0.04	119	0.06	334	0.12	591	0.23	29	0.01	35	0.01	33	0.03	48	0.05
G DATA AntiVirusKit	156	0.07	779	0.38	347	0.13	366	0.14	158	0.10	163	0.10	96	0.12	99	0.13
Grisoft AVG	17	0.01	N/A	N/A	133	0.05	N/A	N/A	29	0.01	N/A	N/A	16	0.00	N/A	N/A
Ikarus Virus Utilities	131	0.06	341	0.17	232	0.08	237	0.09	54	0.03	56	0.03	86	0.11	89	0.11
Kaspersky Anti-Virus	20	0.01	158	0.08	190	0.07	209	0.08	77	0.04	92	0.05	52	0.06	77	0.09
Kingsoft Internet Security	39	0.02	N/A	N/A	141	0.05	N/A	N/A	80	0.04	N/A	N/A	88	0.11	N/A	N/A
McAfee VirusScan	30	0.01	304	0.15	330	0.12	330	0.12	57	0.03	56	0.03	61	0.07	62	0.07
Microsoft Forefront	27	0.01	N/A	N/A	218	0.08	N/A	N/A	63	0.03	N/A	N/A	46	0.05	N/A	N/A
Microworld eScan	764	0.37	764	0.37	371	0.14	371	0.14	179	0.11	179	0.11	158	0.21	158	0.21
Sophos Anti-Virus	24	0.01	559	0.27	212	0.08	228	0.08	39	0.02	53	0.03	33	0.03	60	0.07
Symantec AntiVirus	16	0.01	N/A	N/A	135	0.05	N/A	N/A	36	0.01	N/A	N/A	34	0.03	N/A	N/A
Trend Micro Client Server Security	68	0.03	77	0.04	200	0.07	203	0.07	67	0.03	69	0.04	40	0.04	51	0.05
Trend Micro OfficeScan	67	0.03	69	0.03	182	0.07	183	0.07	51	0.02	52	0.02	48	0.05	52	0.06
Trend Micro PC-cillin	26	0.01	219	0.11	153	0.05	172	0.06	36	0.01	49	0.02	34	0.03	45	0.05
VirusBuster VirusBuster	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Microsoft Forefront Client Security 1.5.1937.0

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	99.10%	**Macro**	100.00%
Polymorphic	96.30%	**False positives**	0

Microsoft's corporate security product is designed to be installed and managed from a central server, the requirements for which run beyond the space provided for this review, but standalone running is available, albeit with a rather unusual, almost silent installation process.

vb 100 VIRUS virusbtn.com Aug 2007

Once up and running, *Forefront* has a fairly simple, pared-down interface, with most of the configuration presumably left for the centralized control utility. Unsurprisingly, it looks much like a part of the operating system, and does its job quietly and efficiently. Speeds were decent, and detection pretty good, with recent efforts to improve coverage of the *VB* test sets paying off and leaving little unidentified.

A strange issue with a single item in the WildList, for which the default setting appears to be to allow it to run when detected, was spotted in the previous test (see *VB*, June 2007, p.10) and still seems to be in evidence. However, on a second run through, with on-demand scanning performed before on-access scanning, the application of the on-demand actions seemed to change things and access to the file was subsequently blocked. The file was invariably detected however, and with nothing in the WildList missed, and no false positives, *Forefront* is deemed worthy of its second VB100 award.

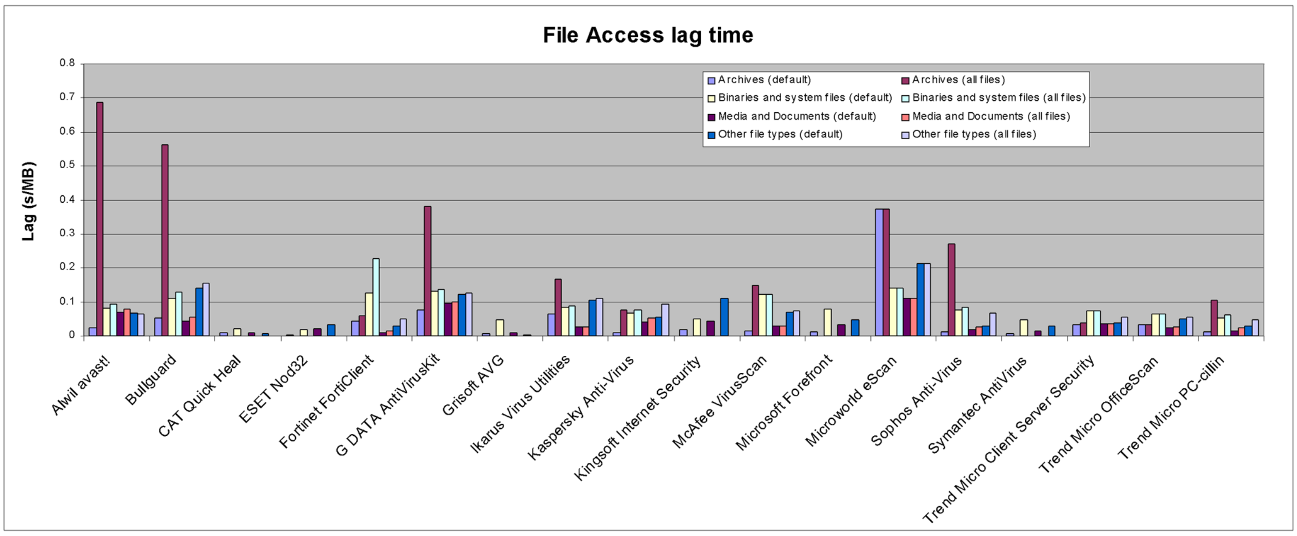
Microworld eScan Internet Security for Windows 9.0.722.1

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

Microworld's *eScan* boasts of being 'powered by *Kaspersky*', and adds a few treats of its own to the hired-in malware scanning. The installer offers a 'Lite' version of a management interface, designed to manage several systems on a small network, while population of an anti-spam whitelist is offered along with the other setup tasks, which include an automatic attempt to update.

vb 100 VIRUS virusbtn.com Aug 2007

As an interface to the *Kaspersky* engine, *eScan* has a few issues on this platform; the popup 'are you sure?' queries are in evidence each time the product is run, and even more



intrusively each time a right-click scan is attempted, with the black screen that precedes many of these popups causing regular moments of concern. The right-click scans themselves seemed not to work at all, unless they run silently and with no logging, which would be of little use to most users.

Scanning from the main interface did work however, and the tests were conducted in this manner, with some slowish speed times reflecting the depth of scanning going on. The expected thoroughness of detection was mostly in evidence, although a handful of macro samples were rather inexplicably missed on access. Nothing was missed in the WildList test set however, and without false positives and with the rogue keylogger left in the clean set spotted, *eScan* battles through to achieve another VB100 award.

Sophos Anti-Virus 7.0.0

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	99.80%
Polymorphic	100.00%	**False positives**	0

Like *Kaspersky*, *Sophos* has upgraded from version 6 to version 7 for this test, a change coinciding with the company's acquisition of a NAC provider. The change in version has had a far less dramatic effect on the product's interface however, which remains pretty much as it was. Once installed, the running of the software is once again impeded by *Vista* warning popups, which (again) frustratingly extend to each time a scan is run from the context menu option. This little annoyance aside, configuration remains flexible in-depth, detection and speeds very good throughout, with a few items in the clean set adjudged risky to corporate networks and the keylogger tool flagged as a possible trojan.

With no misses or false positives *Sophos* also proves worthy of a VB100 award this month.

Symantec AntiVirus 10.2.0.298

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	100.00%
File infector	100.00%	**Macro**	100.00%
Polymorphic	100.00%	**False positives**	0

Symantec's corporate product was one of few to require its installer to be run with full administrator rights, but less complex tweaking of the user access controls was required this time than in the earlier *Vista* test, and soon another familiar interface presented itself for testing.

With the help of this familiarity, navigating the controls was a simple task, with the available options plentiful and accessible, and tests were run through without excessive difficulty. Scores were impeccable, including detection of the keylogger, and speeds were reasonable across the board; without a false positive or missed item of malware, *Symantec* earns itself yet another VB100 award.

Trend Micro Client Server Security for SMB 7.6.1095

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	99.47%
File infector	99.24%	**Macro**	99.68%
Polymorphic	93.29%	**False positives**	1

Trend saw fit to submit no fewer than three separate products for this month's test, the first of which is new to the *VB* test bench. The small business product is a notch below *OfficeScan* in the size of networks it is designed to manage, but operates in a similar way, with a central management console controlling desktop installations, and a simpler interface at the local system level.

Installing the product was simpler than previous experiences with *OfficeScan* had led me to fear – the whole thing installed on the local system, integrating the management tools with a local install in a single go. However, the installation process was not quite done with when, after several setup stages and a query from the *Vista* firewall about whether I wanted to allow the product's *Apache* server to start up, I got access to the interface itself, or rather themselves. For the first 20 minutes or so of using the product, searching for the option which would allow me to control the configuration of scanning and on-access protection from the simpler local console, the browser-based GUI was plagued with blocks from *IE7*'s security measures. *IE7* prompted initially that I should not visit the interface as its certificate was unrecognized, and several times required permission to install the many *ActiveX* controls used by various parts of the system.

When the controls were finally fully running, and control privileges passed to the local user, things became a lot easier, and I found the main interface itself fairly usable and responsive. On-demand scans, which can only be run over the full machine from the 'remote' interface, zipped through nicely, showing fairly solid detection levels, and I soon moved on to the on-access side of things. A repeat of earlier experiences with suspiciously fast run times over the clean sets ensued. After much checking and tweaking of options, I finally found that the on-access scanner was not being sparked by the basic opener tool used for the speed tests, and resorted to copying the collections from one drive to another to obtain detection results.

With the test sets moved to the C: drive, a rerun of the opener provided an odd result – detection was sparked by files being accessed in this manner, but only on the system drive. Unsure whether this was a performance-enhancing function or a bug, I queried it with the developers, who are investigating this oddity, but fortunately the discovery allowed the on-access detection and speed tests to be properly carried out, albeit in a slightly different location from that used elsewhere.

A question arose as to whether not spotting the malware on other drives constituted a failure to detect, but was quickly brushed aside with the justification that many products need a little coaxing to produce on-access results. Unluckily for *Trend*, however, a single item from the batch recently added to the clean set – a development tool provided by *Microsoft* – was falsely identified as spyware, spoiling *Trend*'s chances of a VB100 hat-trick before things had even got going.

Trend Micro OfficeScan 8.0

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	98.39%
File infector	97.80%	**Macro**	99.68%
Polymorphic	93.29%	**False positives**	1

OfficeScan is the big brother of the previous product, similar in design but clearly aimed at much larger organizations. In previous tests on 64-bit platforms, it has been necessary to set up a separate server machine on a 32-bit system and install to the test machine across the network. This proved unnecessary this time, with the management unit installing happily on the *Vista* system alongside the client.

The platform is not officially supported by *Trend*, but it worked well enough to perform the tasks previously deduced as necessary, delegating power to the local client to tweak settings as required. On-access scanning could not, without some complex fiddling, be entirely deactivated from the local console, but beyond that most of my needs were met by the smaller, nimbler interface.

Logging was the only issue which could not be thus circumvented, and a possible indicator of the management interface's unsupported status emerged when trying to save the logs from there, finding them to be somewhat truncated. Lacking the time and resources to set up a fully functioning management server, I made do with running several smaller scans and tagging the logs together. On-demand detection results were similar to those previously spotted, although some of the more venerable samples seemed to be missed, alongside a few other differences which can be accounted for by minor alterations to the defaults.

The problem with on-access scanning on other drives recurred, and further tests showed full detection when moving samples about between drives and writing them in across the network. WildList results were solid, but again a single false positive was raised by the spyware side of the product on access, denying *OfficeScan* its VB100 this time.

Trend Micro PC-cillin Internet Security 2007 15.30.1239

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	99.47%
File infector	99.24%	**Macro**	99.68%
Polymorphic	93.29%	**False positives**	1

On to the home-user product, and one whose interface does not require the installation of an endless stream of *ActiveX* controls. *PC-cillin* is a much more pleasant product to test, being aimed squarely at the home user rather than a corporate admin with a complex security policy to administer and plenty of time to get things set up.

The installation process and interface are simple and pleasant, with curvy lines and plenty of colour to keep the user on side, and while some options are lacking there is still plenty of tweaking available for those who need it. The most obvious shortcoming in all of the *Trend* submissions this month, as pointed out in a recent review of *PC-cillin*, is the lack of a right-click scanning option.

Once again everything went well on demand, and fell over somewhat on access. Scanning was once more most easily achieved on the C: drive, with copying around the system blocked but not, rather worryingly, when copying from a network share onto a local drive, even the system partition. Once again that single clean file was flagged on access, the on-demand virus scan not making use of the spyware engine, and despite decent detection rates *Trend* loses out on the chance of a VB100 award from its three submissions.

VirusBuster VirusBuster Professional 6.0

ItW	100.00%	**Worms & bots**	100.00%
ItW (o/a)	100.00%	**DOS**	99.77%
File infector	99.28%	**Macro**	100.00%
Polymorphic	88.22%	**False positives**	1

VirusBuster’s product is another that seems barely altered over its many appearances on the *VB* test bench, and the familiar installation ran through almost on auto-pilot. Setting up scans, run from the interface in the absence of a right-click option, remains a little taxing despite much practice. However, on-demand tests were soon out of the way – at least until I tried to save the log. Admittedly the log had grown to quite some size during the scanning of the infected sets but saving it nevertheless took an enormous amount of time – during which the interface was unusable.

On-access scanning again proved somewhat problematic, with the opener tool, usually perfectly adequate to test the product, not sparking any detection. Results were once again obtained by copying files around, which meant no comparable speed times could be obtained for this mode. Checking the results showed very good detection though, with nothing vital missed; however, several items in the clean set were flagged with the phrase ‘exploit: attempt to crash system by archive’. While this alert was labelled an error rather than a malware warning, it seems severe enough also to count as a false alarm, which would lead users to delete valid files in the belief that they were some form of attack. The need to make a judgement on this difficult issue was postponed for another day, however, when an item in another part of the set was clearly labelled a virus, and *VirusBuster* thus misses out on an award.

CONCLUSIONS

With few additions to the WildList, and many items removed, the target for the VB100 seemed much easier to achieve this month. However, the rough terrain provided by the platform tripped up several products, with many suffering frustrations imposed by the locking-down of the operating system and others showing idiosyncrasies in their integration into it. On-access scanning, perhaps unsurprisingly, proved difficult to get right for many, while on-demand detection was barely affected by the change of setting.

Almost all of this month’s failures were due to false positives, thanks in part to an enlargement of the clean test sets. The amount of data added was fairly trivial however, with perhaps 100 applications and their component parts added, a minute quantity in comparison with the vast amounts of software in use around the world. These were all fairly common items, mostly taken from the ‘most popular’ lists of several major free and free-trial download sites, and the resultant surge in false detections seems to indicate a fairly significant problem for anti-malware software.

The VB100 rules regarding false positives were changed for this test, with the ‘suspicious’ alert, which in earlier tests allowed products to warn of vague doubts about an item’s intentions without penalty, now limited to covering only correct identifications of genuinely risky software. Hardly any of the products which failed this test did so entirely as a result of this change, but it has made an impact on the results for a few products.

Vista is fairly certain to be a major part of the future of computing, and x64 is also a growing trend with significantly more widespread uptake likely. While security vendors should hopefully be able to hone their wares to operate more smoothly and reliably on the platform before it becomes ubiquitous, it seems unlikely in these times of increasing reliance on heuristics that the false positive will ever be entirely eradicated. We must hope that, for the sake of user confidence in their security products, they can at least be kept to a minimum.

Technical details

Tests were run on identical machines with *AMD Athlon64 3800+* dual core processors, 1GB RAM, 40GB and 200 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy drive, all running *Microsoft Windows Vista x64 Business Edition.*

END NOTES & NEWS

The 16th USENIX Security Symposium takes place 6–10 August 2007 in Boston, MA, USA. A training program will be followed by a two-and-a-half day technical program, which will include refereed papers, invited talks, work-in-progress reports, panel discussions, and birds-of-a-feather sessions. For details see http://www.usenix.org/events/sec07/.

HITBSecConf2007 Malaysia will be held 3–6 September 2007 in Kuala Lumpur, Malaysia. For more details see http://conference.hackinthebox.org/.

SecureDüsseldorf takes place 11 September 2007 in Düsseldorf, Germany. The conference will focus on privacy issues. For further information and registration see https://www.isc2.org/.

Infosecurity New York will be held 11–12 September 2007 in New York, NY, USA. For details see http://www.infosecurityevent.com/.

The 17th International VB Conference, VB2007, takes place 19–21 September 2007 in Vienna, Austria. For the full conference programme including abstracts for all papers and online registration see http://www.virusbtn.com/conference/.

COSAC 2007, the 14th International Computer Security Forum, will take place 23–27 September 2007 in Naas, Republic of Ireland. See http://www.cosac.net/.

The SecureLondon business continuity planning 101 workshop will be held 2 October 2007 in London, UK. For further information and registration see https://www.isc2.org/.

The APWG eCrime Researchers Summit takes place 4–5 October 2007 in Pittsburgh, PA, USA. Academic researchers, security practitioners, and law enforcement representatives will meet to discuss all aspects of electronic crime and ways to combat it. For more information see http://www.antiphishing.org/ecrimeresearch/index.html.

RSA Conference Europe 2007 takes place 22–24 October 2007 in London, UK. See http://www.rsaconference.com/2007/europe/.

Black Hat Japan, takes place 23–26 October 2007 in Tokyo, Japan. Online registration is now open. Call for papers closes 15 August 2006. See http://www.blackhat.com/.

The CSI 34th Annual Computer Security Conference will be held 5–7 November 2007 in Washington, D.C., USA. The conference program and registration will be available in August. See http://www.csi34th.com/.

E-Security 2007 Expo & Forum will be held 20–22 November 2007 in Kuala Lumpur, Malaysia. For event details and registration see http://www.esecurity2007.com/.

The Chief Security Officer (CSO) Summit 2007 will take place 28–30 November 2007 in Amsterdam, the Netherlands. The summit, entitled 'Security strategy to steer your business', offers participants the opportunity to tackle fraud management challenges in a hassle-free environment, surrounded by colleagues. A speaker panel will share direct experiences, successes, and tips gained from managing successful security projects. For details see http://www.mistieurope.com/.

AVAR 2007 will take place 29–30 November 2007 in Seoul, Korea. This year's conference marks the 10th anniversary of the Association of Anti Virus Asia Researchers (AVAR). Enquiries relating to any form of participation should be sent to avar2007@aavar.org.

RSA Conference 2008 takes place 7–11 April 2008 in San Francisco, CA, USA. Online registration will be available from 1 September 2007. See http://www.rsaconference.com/2008/US/.

Black Hat DC 2008 will be held 11–14 February 2008 in Washington, DC, USA. Other 2008 dates include Black Hat Europe 2008, which takes place 25–28 March 2008 in Amsterdam, the Netherlands, and Black Hat USA 2008, which takes place 2–7 August 2008 in Las Vegas, NV, USA. For details see http://www.blackhat.com/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:

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

Tel: +44 (0)1235 555139 Fax: +44 (0)1235 531889

Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

NEWS & EVENTS

SPAMMERS SENTENCED

The first US spammer to be convicted under the 2003 CAN-SPAM Act was sentenced last month, nearly three years after agreeing a plea deal with federal prosecutors. In September 2004, 40-year-old Californian Nicholas Tombros became the first person to be convicted under the CAN-SPAM Act when he pleaded guilty to sending spam messages advertising pornographic websites. He admitted to having driven around the Los Angeles neighbourhood of Venice searching for unsecured wireless networks which he then used to send the spam messages – having obtained the email addresses from a credit card aggregation company of which he was a former employee.

Tombros was sentenced last month to three years' probation, and six months' home detention, as well as ordered to pay a $10,000 fine. The reason for the long delay in sentencing was not disclosed.

Meanwhile, Australian mobile phone marketing company *DC Marketing Europe* has been fined almost AU$150,000 by the Australian Communications and Media Authority for breaching the country's Spam Act in July and August 2006. The company – notorious for its 'missed call' marketing schemes – was charged with sending unsolicited messages that failed to identify the sender and did not allow the recipient to unsubscribe.

EVENTS

CEAS 2007 takes place 2–3 August 2007 in Mountain View, CA, USA. For details see http://www.ceas.cc/.

The 11th general meeting of the Messaging Anti-Abuse Working Group (MAAWG) will take place 8–10 October 2007 in Washington D.C., USA. See http://www.maawg.org/.

TREC 2007 will be held 6–9 November 2007 at NIST, MD, USA. See http://plg.uwaterloo.ca/~gvcormac/spam.

FEATURE

EMAIL SENDER AUTHENTICATION: ADVANTAGES AND SHORTCOMINGS

Alberto Treviño and J. J. Ekstrom, PhD.
Brigham Young University, USA

The severity of the spam and malware calamity should not be new to readers of this publication; neither should the close relationship between the two, nor the difficulty in combating them. For new readers, things can be summed up as follows: the situation is perilous, spam and malware go hand in hand, and defence is difficult and expensive. It often seems we are in an arms race, with each side trying to out-gun their opponent. Although we all secretly hope for and seek the magic weapon that will obliterate the problem once and for all, the reality is such a weapon does not currently exist, and we still rely on our tried and tested methods of defence to help us survive.

EMAIL AUTHENTICATION

During the last few years we have seen the emergence of several technologies commonly referred to as 'email authentication'. Email authentication attempts to verify and certify the authenticity of an email. Email authentication is usually divided into two categories [1]:

1. *Sender authentication* attempts to verify the validity of an email's source. More specifically, it attempts to detect header forgeries and ensures that emails from a particular domain are sent from trusted systems. Well-known specifications in this category include Sender Policy Framework (SPF) and SenderID.
2. *Content authentication* ensures that the contents of an email have not been tampered with or modified in any way during delivery. A well-known specification in this category is DomainKeys, which recently received RFC standard status.

This paper will focus mainly on sender authentication.

PROS AND CONS OF EMAIL VERIFICATION

In reality it is hard to find a multi-purpose tool that is the best at everything. In order to use a tool effectively, it is essential to understand its strengths and weaknesses. The same applies to email authentication. In order to use it effectively, it is necessary to understand what it can do well, what it can't do very well, and what it can't do at all.

To understand properly the best application of email authentication technologies let's begin with their deficiencies:

- Email authentication is not a silver bullet. It will not eliminate spam once and for all and it will certainly not do it overnight.
- It will not stop spammers from attempting to spew spam.
- It will not tell you directly if a message is spam or ham.
- Spammers can also make use of these technologies and 'authenticate' in the same way as legitimate email.

These deficiencies have been the cause of heavy criticism from some in the internet community. Several have even labelled authentication technologies as 'useless'. We disagree. If used properly, email authentication can help *reduce* (not eliminate) spam.

PROPER USE OF EMAIL AUTHENTICATION

So how do you use email authentication properly? You use it to help establish trust. For example, sender authentication helps answer the questions 'is this email coming from the place it should come from?' If the sender authentication fails, we have reason to doubt the legitimacy of the email. But if the authentication is successful it won't automatically mean the email is legitimate.

In other words, failed authentication does not make an email spam and successful authentication does not make it ham. This is the area where many have problems understanding the proper use of email authentication. They erroneously believe that authentication by default equates to legitimacy. To their credit, spammers were quick to point out this flawed idea. Unfortunately, many who made this mistake were quick to blame the technology rather than their own misunderstanding of the technology for its failures and precipitately dubbed email authentication useless.

So what does email authentication do for us? Let's begin with sender authentication in the form of SPF and its cousin SenderID.

SPF publishes rules via DNS records about which SMTP servers are allowed to send email for a particular domain. If we receive an email from a server that does not conform to the published rules, the possibility of header forgery for that email increases. SPF (if not set up too broadly) effectively reduces the number of SMTP servers that can send email for a given domain from hundreds of millions to a mere handful. This can help reduce the impact of botnets and help catch phishing attacks.

Message authentication can also help authenticate email by providing a verifiable signature for each email message. This helps us determine whether the message was signed by a valid server and whether the message has arrived unmodified.

SENDER AUTHENTICATION WITHOUT SPF OR SENDERID

One major shortcoming of sender authentication technologies is that they are not currently mandated. Not all domains have deployed these technologies and they don't enjoy broad deployment. Politics have also played a role, with many early adopters taking sides with one technology or the other.

During the MIT Spam Conference 2007 [2] we presented a technique similar to SPF that uses existing DNS records to authenticate email. We called this technique 'Relay Detection' [3]. It works under the assumption that legitimate email is rarely relayed these days, and that spammers continue to relay spam, mostly through botnets. In other words, a *Hotmail* user will send email through *Hotmail* servers while a spammer claiming to come from *Hotmail* will use an open relay or a machine in a botnet to send the email.

The first step in Relay Detection is to verify the HELO identity of the sender SMTP, either during the SMTP

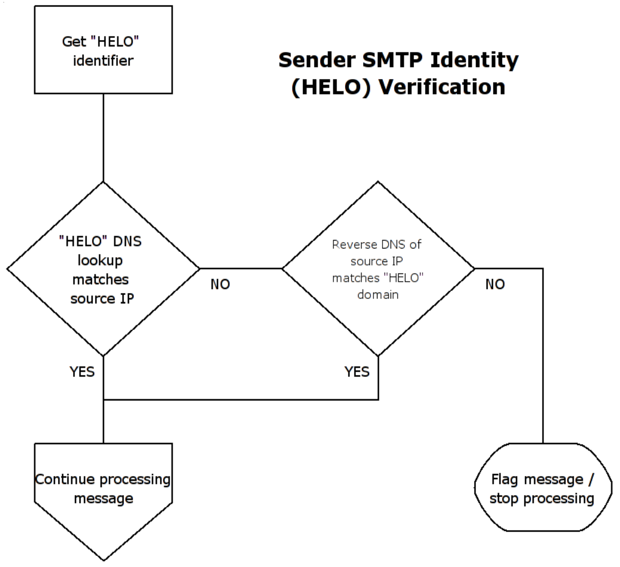

Get "HELO" identifier
Sender SMTP Identity (HELO) Verification
"HELO" DNS lookup matches source IP
NO
Reverse DNS of source IP matches "HELO" domain
NO
YES
YES
Continue processing message
Flag message / stop processing

Figure 1: Sender SMTP identity verification.

transmission or through the correct Received header (see Figure 1). The SMTP standards indicate the sender SMTP should send its Fully Qualified Domain Name (FQDN). Therefore we verify the validity of the HELO identity as follows:

- Verify it is a FQDN.
- Verify it against existing DNS entries by performing a *forward* DNS lookup and comparing the DNS record's IP address to the sender SMTP IP address.
- As a last resort, perform a *reverse* DNS lookup on the sender SMTP IP address and compare the two domain names for consistency.

In our preliminary testing, these simple tests were capable of detecting over 60% of spam with less than 0.35% false positives. Although these numbers may not be terribly impressive, they can be extremely useful if these checks are implemented at the SMTP level *before* the email is even received.

The second step in Relay Detection is to verify the second command in the SMTP transmission: MAIL FROM (see Figure 2). If done after the SMTP transmission, you can use the Return-Path header. The main goal of this test will be to establish through DNS records a relationship between the domain information given in the HELO command with the domain in the MAIL FROM address. You can verify the validity of the MAIL FROM address as follows:

- Extract the domain part of the email address (hotmail.com, gmail.com, etc.).
- Compare the domain in the email address with the domain in the HELO identity. If they match (e.g. hotmail.com and mx4.hotmail.com would match), the verification passes.
- If the domains don't match, you perform an MX DNS lookup on the email domain. If any domain in the MX record matches the HELO identity domain, the verification passes.
- If the MX and HELO domains don't match, you perform a *forward* DNS lookup on the MX records to obtain their IP address. Then you compare the MX record IP addresses with the address of the sender SMTP. If the IP addresses are in the same IP address range (even in the same set of /16 is OK), the verification passes.
- As a last resort, you can perform a DNS lookup on the email address domain and see if any of the IP addresses are in the same IP address range as the sender SMTP.

In our preliminary testing we were able to identify 78% of the spam messages not flagged by the Sender SMTP Verification tests with roughly a 0.5% false positive rate. Overall, these two forms of verification flagged over 90% of spam with a less than 1% false positive rate.

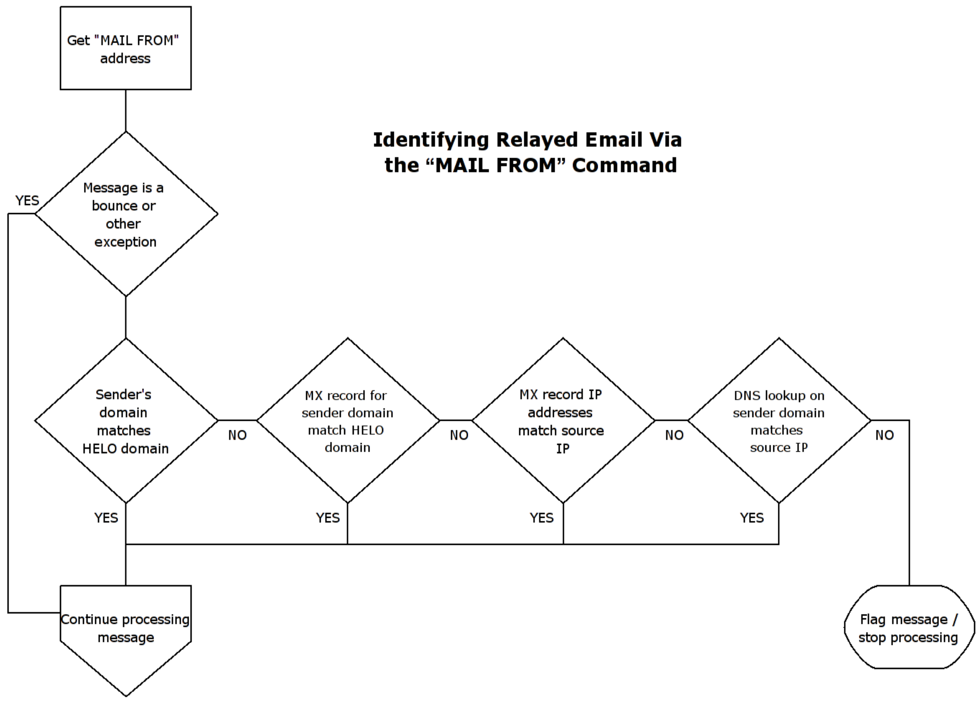

Identifying Relayed Email Via the "MAIL FROM" Command
Get "MAIL FROM" address
Message is a bounce or other exception
YES
Sender's domain matches HELO domain
NO
YES
MX record for sender domain match HELO domain
NO
YES
MX record IP addresses match source IP
NO
YES
DNS lookup on sender domain matches source IP
NO
YES
Continue processing message
Flag message / stop processing

Figure 2: MAIL FROM verification.

Be aware that the tests and their order are not arbitrary. They are designed to work for nearly all email configurations, including domains in virtual hosting environments.

The only cases where these tests fail is when headers are forged or email is relayed (after all, that's what they are looking for). However, these tests are usually very fast and consume very little processing time, with most of the time being spent waiting for DNS queries.

HOW SENDER AUTHENTICATION CAN HELP REDUCE SPAM

It is important to note that sender authentication technologies have been adopted by many spammers. This greatly reduces the efficacy of email authentication as a spam-fighting tool.

Many who have encountered this unfortunate fact have concluded that these technologies simply ‘don’t work’ and are not worth the effort. Luckily, email authentication is still helpful in the fight against spam. Just because a screwdriver doesn’t make a good hammer doesn’t mean it should be thrown out.

As we mentioned earlier, email authentication helps the initial establishment of trust. Sender authentication in any or all of its flavours does one thing very well: it reduces the number of servers that can legitimately send email for a domain to a privileged few.

If used properly, sender authentication prevents header forgeries, forces spammers to be more honest and pushes them to start using their own domains. This has important positive implications for mitigating phishing attacks. For example, a scam artist trying to impersonate a bank (such as *Bank of America*) cannot simply forge the email headers any more.

In order to bypass these email authentication techniques the spammer would have to:

1. Set up a new domain.
2. Set up all proper DNS records, including MX records for the machines in the botnet.
3. Set up SPF/Sender ID records.

Not only does this start to affect the economics of spam, it also plants the seed of distrust for the receiver. An astute recipient receiving an email from ‘Bank of America Support’ <bankofamerica@qqpdf.com> may wonder why the email did not originate from bankofamerica.com. Not all email users may be so savvy but anti-phishing techniques like those used by modern browsers could also help diffuse the attack.

CONCLUSIONS AND FUTURE WORK

Although our research may not be seen as revolutionary, it has helped us confirm several things:

1. Many spammers continue to relay email.
2. Simple header analysis combined with DNS lookups can be used to authenticate an email’s sender.
3. Sender authentication (if used properly) can still help reduce spam.
4. The only way for spammers to circumvent any form of email authentication is to be more honest and comply with existing and emerging standards.
5. Email authentication technologies work. They may not eliminate all spam, but they can help eliminate some of the spam.

In continuing with our research we may investigate further the relationship that could be established between sender identification and domain trust.

With sender authentication technologies helping verify the source of incoming email, we can more fully develop trust relationships between domains.

With the help of other spam-filtering technologies and the confidence that an email is coming from a verified source, it should be possible to measure a domain’s tendency to spew spam. Those measurements could be calculated automatically from incoming email and could provide another measure to help determine the probability of a message being spam. Even though several companies specialize in the analysis of spam for the purposes of publishing blacklists and such, this technique may improve blacklist and whitelist efficiency by ensuring the lists are relevant to the spam and legitimate email received by a particular host.

Another possible direction for our research is to use sender authentication as a means of prioritizing spam filtering, virus scanning and delivery queues. Along with the domain spam measurements mentioned above, incoming SMTP connections could be prioritized and handled appropriately, allowing efficient routing. For example, an untrusted domain could be greeted by a tar pit in an effort to slow the spammer and wait for the sender to give up. At the same time messages from another domain with a good ham record but poor virus record could immediately be sent through a very thorough virus scanner. And, in some cases, it may even be possible that some domains could bypass spam filtering and virus scanning altogether and be queued for immediate delivery.

These ideas are still in their early infancy. We are barely exploring the many possibilities provided by sender authentication. In either case, we feel confident that email authentication (both sender authentication and message authentication) can help reduce email *if* it is used as a tool for establishing trust and not mistaken as a complete spam solution in and of itself.

ENDNOTES

[1] http://www.openspf.org/Related_Solutions.

[2] http://www.spamconference.org/.

[3] For the full paper and presentation follow-up see http://mel.byu.edu/spam/.