MAY 2011

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

UNRUY UPGRADED

Win32/Unruy.AD conceals its code at a low level and infects the Master Boot Record so that it will be loaded before the operating system (and any security solutions installed by the user) have a chance to kick in. Robert Lipovský and Peter Hlavatý provide all the details.

page 4

SOMETHING FOR NOTHING

Free anti-virus continues to be all the rage, with more and more firms jumping on the bandwagon. John Hawes takes a look at one of the latest offerings: the free version of Agnitum's Outpost Security Suite, and finds that you can get quite a lot for nothing these days.

page 17

VBSPAM CERTIFICATION

The 13th VBSpam test didn't prove unlucky for any of the participating products, with all 19 entrants receiving a VBSpam award. Martijn Grooten presents the detailed figures that distinguish the good products from the *really* good ones, and takes a look at some additional features of the products on test.

page 21

'... with the facts laid bare, the cross-border cyber enforcement issues between nations will be discussed differently.'

Wout de Natris
De Natris Consult

A 'DEFENCE TRIANGLE'

Discussing the fight against spam, malware and cybercrime has become almost a national pastime. The vulnerabilities of the Internet and networks are such that some believe that our existence as we know it could be threatened by a single keystroke. Whether or not that is the case, the level of intelligence relating to cybercrime needs to be improved in order to prioritize defence. In the following I will make some suggestions to achieve just that.

I propose the introduction of a 'defence triangle'. At its corners are: CERTs and anti-abuse desks, anti-spam enforcement and anti-cybercrime enforcement. The intelligence position of each corner can be strengthened.

It's a fact that most countries (if not all) have no central record of anything relating to cybercrime. For convincing figures about cybercrime we need to look to AV vendors and organizations like *Spamhaus*, but their statistics do not necessarily cover the whole range of incidents. In order to be able to prioritize correctly, one needs reliable data.

It is safe to assume that the CERTs have reliable data on security breaches, botnets and such (if they catch the threat). That leaves the other two corners of the triangle. I propose the building of two central databases to which members of the public can report incidents online.

One for spam, phishing, any suspicious looking emails and malware, and one for other types of cybercrime. Analysis of this data would give the law enforcement community a tremendous boost in intelligence and threat assessment and avoid the need to use vendor-supplied (thus commercially driven) data.

So we have central databases, but we still need industry and institutions to commit to the fight against cybercrime by reporting cybersecurity incidents to the proper authorities. Are incidents actually being reported? How can cyber priorities be set if intelligence breaches, phishing and extortion are not being reported? The reporting of these crimes might help to prevent panic when/if a serious breach occurs. Everyone concerned – including politicians and policy makers – would already be aware of and prepared for such incidents.

To raise the level of intelligence relating to cybercrime the three partners of the triangle must cooperate. Exchange of reliable data must be the first step. Through interaction and coordination, each of the partners can focus on direct and verifiable threats.

Of course, none of this will happen magically. Governments must provide the conditions in which the often conflicting interests of industry, security and privacy are brought together and turned into a positive force. At a minimum this will be a facilitating role, but would most likely also need to be a financial, and potentially steering role.

I foresee three initial steps:

1. Countries set up national online incident report databases, which feed into an analysis and coordination centre.
2. Industry and other institutions report cyber incidents to the proper authorities.
3. Governments provide the conditions for coordination and cooperation between criminal and so-called 'softer' law enforcers, CERTS and industry.

Through these steps reliable data will become available and all involved will be able to prioritize towards dealing with the most acute cases, whether in national security or cybercrime (related) issues. The ensuing coordinated actions will drive back crime on the Internet, enable more criminals to be caught, and make the Internet environment safer. I even believe that with the facts laid bare, the cross-border cyber enforcement issues between nations will be discussed differently. In theory, it doesn't seem that hard, but who will be willing to pick up these challenges?

NEWS

CRIMEWARE KIT RELEASED FOR MAC

A new crimeware kit has been discovered that looks set to bring trouble for *Mac OS X* users. The first known crimeware kit aimed specifically at the *Mac OS X* platform was released recently on underground forums, according to Danish IT security firm *CSIS Security Group*.

The kit, which is advertised as the Weyland-Yutani BOT and costs $1,000, currently supports web injects and form grabbing in *Firefox*, with its creators promising the same functionality for the *Chrome* and *Safari* browsers in the near future. The webinjects templates are the same as those used in the very popular *Windows* crimeware kits Zeus and Spyeye.

Crimeware kits have become a ubiquitous part of the *Windows* malware scene in the last few years, allowing users to create their own custom versions of malicious software that can turn machines into remotely controlled bots and/or harvest data from the infected machines.

The DIY kit's developers have indicated that they also plan to release kits for *iPads* and *Linux* machines.

With many *Mac* users still convinced that the platform is more secure than *Windows* they will need to be on their guard against socially engineered attacks – a need reaffirmed by the recent discovery of rogueware (or fake AV) targeting *Mac* users. A recent surge of SEO poisoning attacks on *Google* (hijacking search results of queries ranging from global warming to the death of Osama bin Laden) has turned up malicious domains serving two rogueware applications specific to *Mac OS X*: Best Mac Antivirus and MACDefender.

While there is currently significantly less malware in existence for *Mac OS X* than there is for *Windows*, these developments are an indication that criminals are taking an increasing interest in the *Mac* platform.

DNS HACK

Security firm *Cloudmark* has reported the hacking of a server that provides DNS for various legitimate domains. The hackers did not touch the DNS record for the www-subdomain (e.g. www.example.com) – making the hack less likely to be discovered – but instead used DNS wildcards to make any other subdomain (e.g. ww.example.com or jhjkh.example.com) resolve to their own servers. These subdomains were then used in spam campaigns.

Because most URL and domain blacklists only consider the least significant part of the domain name (in this case example.com), it is less likely that these domains would be blocked by spam filters. *Cloudmark* contacted both the hosting company and the company that provides DNS for the domains in question to alert them to the situation.

Prevalence Table – March 2011[1]

Malware	Type	%
Exploit-misc	Exploit	8.75%
Autorun	Worm	7.25%
Adware-misc	Adware	7.07%
Heuristic/generic	Virus/worm	6.06%
VB	Worm	5.84%
Conficker/Downadup	Worm	5.57%
FakeAlert/Renos	Rogue AV	5.10%
Agent	Trojan	3.50%
Sality	Virus	3.31%
OnlineGames	Trojan	2.65%
Zbot	Trojan	2.31%
StartPage	Trojan	1.97%
PDF	Exploit	1.84%
Crack/Keygen	PU	1.81%
Ircbot	Worm	1.78%
Dialler-Misc	Trojan	1.70%
AutoIt	Trojan	1.67%
Kryptik	Trojan	1.59%
Downloader-misc	Trojan	1.56%
Heuristic/generic	Trojan	1.49%
Crypt	Trojan	1.30%
Injector	Trojan	1.26%
Virut	Virus	1.20%
Delf	Trojan	1.13%
FakeAV-Misc	Rogue AV	1.08%
Dropper-misc	Trojan	1.02%
Tanatos	Worm	0.99%
Bifrose/Pakes	Trojan	0.89%
HackTool	PU	0.88%
Small	Trojan	0.79%
Themida	Packer	0.78%
Iframe	Exploit	0.75%
Others[2]		15.09%
Total		100.00%

[1]Figures compiled from desktop-level detections.

[2]Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

MALWARE ANALYSIS

PUTTING TROJAN.HASHISH OUT TO GRASS

Robert Lipovský, Peter Hlavatý
ESET, Slovakia

Some time ago, we noticed a new malware sample which conceals its code at a low level – in raw sectors of the hard drive – and infects the Master Boot Record so that it will be loaded before the operating system (and any security solutions installed by the user) have a chance to kick in. Whenever we come across such behaviour, the familiar rootkit families Win32/Mebroot and Win32/Olmarik (a.k.a. TDL) come immediately to mind. However, after a closer inspection, we realized that we were looking at something a little different: an upgraded version of the Unruy family of trojan downloaders, which has been around since 2009. We detect this variant of the trojan as Win32/Unruy.AD.

ANALYSIS

The initial malicious executable (which installs the rootkit) is lightly obfuscated and features a few anti-debugging tricks. It also tries to detect and disable some anti-virus programs. The executable is responsible for writing its system drivers to raw sectors of the hard drive and infecting the Master Boot Record (MBR). The installer uses a privilege escalation exploit, which involves setting a specially crafted SystemDefaultEUDCFont value under the registry key HKEY_CURRENT_USER\EUDC and then calling the EnableEUDC() function from GDI32.dll.

An interesting point of note is that the malware has a conflict with the encryption software *TrueCrypt*, if it is active. It checks whether *TrueCrypt* is turned on, and if the part of the drive to which Unruy.AD wants to write its files is encrypted, the installation aborts and the trojan deletes itself. A debug string found in the code shows that the author(s) of the bootkit named it 'Trojan.Hashish' (see Figure 1).

Two more executable files are dropped and executed by the installer. The first is a *Windows* shell batch file (.CMD), which is used to remove the dropper. Verses from The Book of Genesis are included as comments in this file (see Figure 2). The second executable waits for two hours before rebooting the system. The reboot is required for the driver to load fully from the code written to the MBR.

```
.text:00138C24 aDrvirusTrojan_hash db 'DrVirus: Trojan.Hashish detected a debugger!',0Ah,0
.text:00138C24                                         ; DATA XREF: infect_mbr+28↑o
```

Figure 1: Win32/Unruy.AD has named itself 'Trojan.Hashish'.

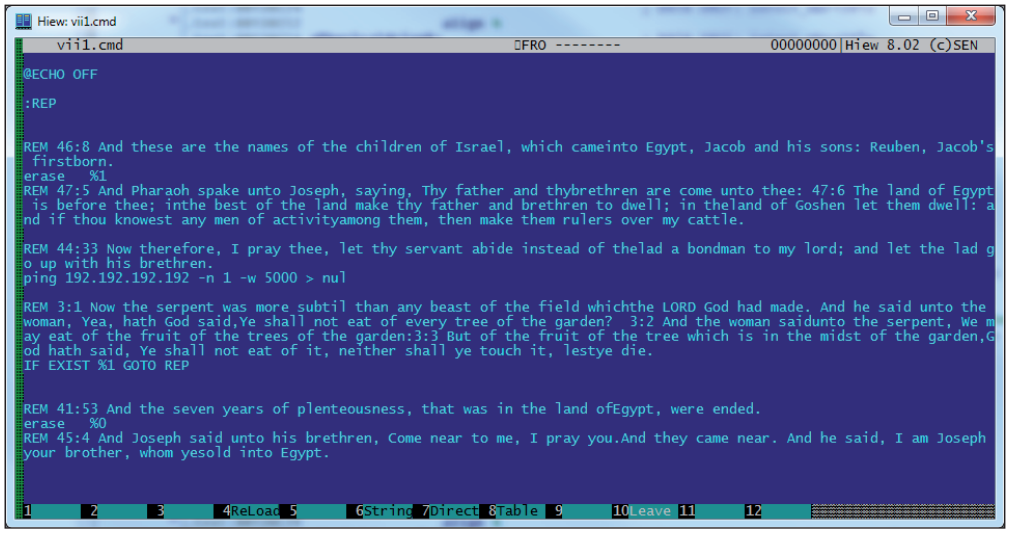


Figure 2: Shell script for removing the original malware executable.

```
seg000:0000810E a?C05f9abbcd50c64876b361456153e39e db '?:\c05f9abbcd50c64876b361456153e39e',0
seg000:00008132 a?15137ef73def24f4f00239628a70df43 db '?:\15137ef73def24f4f00239628a70df43',0
seg000:00008156 a?9d02867239b96bff7d5e78a234aa4955 db '?:\9d02867239b96bff7d5e78a234aa4955',0
seg000:0000817A a?52e647d150652afd4ab057e51945f3e8 db '?:\52e647d150652afd4ab057e51945f3e8',0
```

Figure 3: Links to directories in Win32/Unruy.AD's file system.

The boot sequence of an infected computer consists of several stages, involving the malicious MBR code, 16-bit and 32-bit code, the hooking of int 0x13 and IoInitSystem, and so on. These techniques are obviously inspired by bootkits such as Mebroot. Afterwards, the malware drivers are loaded from the bootkit's file system. The trojan's file system is implemented as pseudo-directories that bypass the *Windows* file system (FAT or NTFS). Unruy.AD references these through links, as seen in Figure 3.

In fact, this is what Olmarik does, though in a slightly different way. An important detail here is that the 'symbolic links' seen in Figure 3 are not

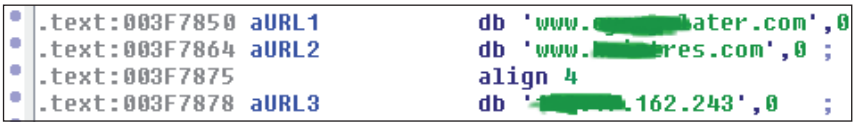


Figure 4: URLs in the backdoor.

Figure 5: URL in the initial installer.

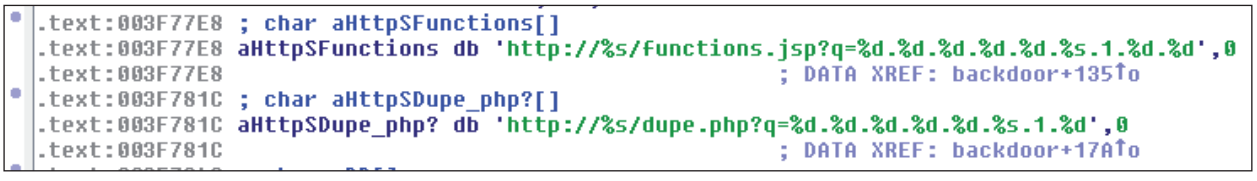


Figure 6: URL format strings.

Figure 7: Spoofed HTTP referrer.

direct links to files (which would suggest that there are four of them), but links to pseudo-directories. The number of drivers depends on the malware version. Unruy.AD includes three.

The trojan uses a data structure for sharing functions between the three drivers and its user-mode components. The functions include the execution of tasks such as writing to the trojan's file system, writing data or code to memory, and so on. The first driver acts as a service dispatcher, providing user-mode code access to these functions via DeviceIoControl.

The second driver creates mutexes of other known malware families (e.g. Win32/Sality), so that they cannot execute – this may indicate competition between different malware authors and gangs.

The third driver is responsible for injecting the payload into user-mode processes. Even this injection is done in several steps, which adds a layer of self-defence. The driver reads the encrypted payload binary from the malware's file system and, along with a loader stub, injects it into services.exe. The code within services.exe creates a thread in svchost.exe, to which it attaches itself as a debugger and passes the payload. This thread finally decrypts and runs the payload binary. Further self-protection techniques prevent the termination of the malware's user-mode processes (svchost.exe and iexplore.exe).

The final payload implements a backdoor which is able to download, store and run files from a remote server. This allows the malware to perform various activities subsequent to infection. Unruy.AD contains three advertisement-related URLs in the backdoor itself (see Figure 4), to which it tries to connect, but the backdoor is also capable of being reconfigured. A fourth advertisement-related URL is contained in the initial installer (Figure 5).

In addition to the ability to download other files from these URLs, the HTTP requests sent by the trojan include information about the infected computer (volume information, *Windows* product ID, computer name and system information) as parameters (see Figure 6).

The trojan has the option to send the requests either via *Internet Explorer* (iexplore.exe) or directly, and to spoof the referrer value (see Figure 7). This behaviour simulates clicks on advertisement banners, which suggests that the trojan is attempting to cheat a pay-per-click business model and/or carry out black hat search engine optimization (SEO).

FUTURE DEVELOPMENT

When analysing the binary, we also noticed some worm-like functionality for distributing the malware across the network – however, these features are inactive in the current release. The current release also lacks rootkit techniques for hiding itself in the infected system – but considering the fact that the malware is under development, and bearing in mind its apparent sources of inspiration, this is likely to change in the near future.

TECHNICAL FEATURE

FLIBI: EVOLUTION

Peter Ferrie
Microsoft, USA

The Flibi virus demonstrated that a virus can carry its own 'genetic code' (see *VB*, March 2011, p.4), and if the codons[1] (the p-code form of the virus), the tRNA[2] (the translator function), or the corresponding amino acids[3] (the native code) are mutated in some way, then interesting behaviours can arise.

Each codon is used as a relative offset into a table of amino acids. There is a single pointer to the table. Mutation of a codon might cause a new amino acid to be produced, since it might now point to a different entry in the table. Mutation of the pointer would almost certainly be fatal since many codons would not be translated into the correct amino acids. Mutation of the amino acid itself might produce new behaviour, depending on the change. For example, a shift could become a rotate.

The virus has the ability to move a sequence of codons to a later position in the stream[4], and then fill the gap with no-operation instructions. In most cases, this simply results in the replacement of the codons at the destination[5]. Of course, if the selected sequence appears at the end of the defined stream (there is a lot of slack space after the last meaningful codon), then the size of the defined stream will increase slightly each time that condition occurs. However, the size of the buffer remains fixed. Therefore, new sequences can only appear when the translator code is modified to increase the number of codons that are translated, thus 'translating' garbage beyond the original end of the stream. That garbage could potentially be modified over time to eventually produce meaningful functionality. Its location in the virus body would change over time as a result of the codon deletion, allowing the new amino acids to 'migrate' to a final position where they become truly useful. The human eye did not spring fully formed from the dust of the earth but was the result of gradual refinements in image accuracy. Something similar can occur here, where the sequence of amino acids does not need to work completely (or even at all) in order to be useful (or just retained). As unlikely as these things are, millions of computer years from now, we might see some of the following transformations.

The aim of this article is to demonstrate how some instructions from the original set might be removed by replacing them with functionally equivalent code sequences using the remaining instructions. One advantage of a smaller instruction set is that it allows an increase in the number of codons that can map to a single amino acid, thus making the body more resilient to corruption. Further, a sequence of instructions has a smaller risk of lethal mutation than a single instruction, because the risk is spread over a wider area.

We begin with a brief overview of the language itself. There are 45 commands in the release version (there were only 43 in the preview version). There are three general-purpose registers ('A', 'B' and 'D', which correspond to the 'eax', 'ebp' and 'edx' CPU registers); one temporary register, upon which all operations are performed (which corresponds to the 'ebx' CPU register); one 'operator' register, which holds the value for any operation that requires a parameter (which corresponds to the 'ecx' CPU register); and two buffer registers, one of which holds the destination for branching instructions (which corresponds to the 'esi' CPU register), and the other holds the destination for write instructions (which corresponds to the 'edi' CPU register).

The language supports the following commands:

- _nopsA, _nopsB, _nopsD, _nopdA, _nopdB, _nopdD
- _saveWrtOff, _saveJmpOff
- _writeByte, _writeDWord
- _save, _addsaved, _subsaved
- _getDO, _getdata, _getEIP
- _push, _pop, _pushall, _popall
- _zer0
- _mul, _div, _shl, _shr, _and, _xor
- _add0001, _add0004, _add0010, _add0040, _add0100, _add0400, _add1000, _add4000, _sub0001
- _nopREAL
- _JnzUp, _JzDown, _JnzDown
- _CallAPILoadLibrary, _CallAPIMessageBox, _CallAPISleep (release version), _call
- _null (release version, it has no actual name)

This set can be reduced in several ways. The most obvious candidates for removal are the three API calls (two in the

[1] A codon is a trinucleotide sequence of DNA or RNA (the nucleic acids that contain the genetic instructions used in the development and functioning of living organisms) that corresponds to a specific amino acid. See http://www.genome.gov/Glossary/index.cfm?id=36.
[2] Transfer RNA, or tRNA, is a small RNA molecule that is involved in protein synthesis. See http://www.wiley.com/college/boyer/0470003790/structure/tRNA/trna_intro.htm.
[3] Amino acids are the building blocks of proteins. See http://en.wikipedia.org/w/index.php?title=Amino_acid&oldid=412676887.
[4] There is a bug in this code, which can result in attempting to copy more bytes than exist in the source.
[5] There is an additional case where the destination is the same as the source, in which case the codons are deleted.

preview version[6]). The APIs can be called using the '_call' command if the API addresses are placed in the data section in this way:

```
_getDO      ;get data offset
_addnnnn    ;adjust ebx as appropriate to reach the
            ;required offset
_call       ;call the API
```

This leaves 42 commands remaining (41 in the preview version).

The '_zer0' command can be removed by using this code:

```
_save   ;ecx = ebx
_xor    ;ebx = 0
```

41 (40) commands now remain.

The '_subsaved' command (which performs the action 'ebx = ebx – ecx') can be removed, and the '_addsaved' command (which performs the action 'ebx = ebx + ecx') can be used instead, with a slight change. Specifically, the new value of the 'ecx' register is '-ecx' (such that 'ebx = ebx + -ecx'). However, there is no negate command, so an equivalent result must be achieved using the combination of operations that perform a 'not' and an 'add 1'. The problem is that a 'not' operation uses the value '0xffffffff', which requires many steps to construct. Given the existing instruction set, it would be simplest to place the value '0xffffffff' in the data section[7]. It must be placed at the start of the data section, because the '_addnnnn' commands can be removed, leaving no way to select another offset. This algorithm can then be used:

```
xor     ebx, 0xffffffff
inc     ebx
```

which we translate into this code:

```
_push
_getDO          ;get data offset
_getdata        ;fetch 0xffffffff
_xor            ;logically 'not' ebx
_add0001        ;increment result to complete negate
_save           ;replace ecx
_pop
_addsaved       ;ebx = ebx + ecx
```

40 (39) commands remain.

In the same way, the '_sub0001' command can be removed by using this code:

```
_push
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_pop
_addsaved       ;ebx = ebx - 1
```

39 (38) commands remain.

The '_addnnnn' commands exist for convenience, but all of the commands apart from '_add0001' can be constructed using the '_add0001' command. Thus, the '_add0004', '_add0010', '_add0040', '_add0100', '_add0400', '_add1000' and '_add4000' commands can be removed.

32 (31) commands remain.

The '_add0001' command can also be removed, because the number '1' can be recovered from the value '0xffffffff' by using this code:

```
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff

;here is a horrible trick:
;modern CPUs limit the shift-count to 0x1f by taking
;the low five bits for the count and simply discarding
;the rest of the value internally, this performs a
;cl & 0x1f and it's exactly what we need

_shr            ;ebx = ebx >> cl
_save           ;ecx = 1
```

From then on, the '_addsaved' command can be used to increment the 'ebx' register as needed.

31 (30) commands remain.

Of course, it would require very many uses of the '_addsaved' command in order to construct large values, but value construction can be accelerated by using the '_shl' and '_xor' commands.

For example, constructing the value '2' is a matter of the following:

```
_shl            ;ebx = ebx << cl (ebx and ecx are '1'
                ;from above)
```

Constructing the value '3', beginning with the 'ebx' and 'ecx' registers holding the value '1', as above, is a matter of the following:

```
_shl            ;ebx = ebx << cl
_xor            ;ebx = ebx ^ ecx
```

Constructing the value '4', beginning with the 'ebx' and 'ecx' registers holding the value '1', as above, is a matter of the following:

```
_shl            ;ebx = ebx << cl
_shl            ;ebx = ebx << cl
```

And so on. Given this algorithm, we can see that the value '0xffffffff' is not the only possible 'base constant'. The value '1' could be used instead, since the value '0xffffffff' could be produced from it in the following way:

[6] The '_CallAPISleep' command was added to the release version because the API resolver code could not resolve the Sleep() API on certain platforms. The reason is described in detail in the previous article (*VB*, March 2011, p.4).

[7] It would be even simpler to introduce an instruction which performs a 'mov ebx, 0xffffffff'.

```
_getDO          ;get data offset
_getdata        ;ebx = 1
_save           ;ecx = 1
_shl            ;ebx = 2
_xor            ;ebx = 3
_shl            ;ebx = 6
_xor            ;ebx = 7
_shl            ;ebx = 0x0e
_xor            ;ebx = 0x0f
... [54 steps]
_shl            ;ebx = 0xfffffffe
_xor            ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
```

Clearly, it is far simpler to go from '0xffffffff' to '1' than the other way around. Note that values can also be constructed using the 'reverse' of this technique, to reduce the number of shifts required. For example, constructing the value '0x80000000' is a matter of the following:

```
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_shl            ;ebx = 0x80000000
```

Constructing the value '0x40000000' is a matter of the following:

```
_push
_shr            ;ebx = ebx >> cl (ebx = 0x80000000,
                ;ecx = 0xffffffff from above)
_save           ;ecx = 1
_pop
_shr            ;ebx = 0x40000000
```

However, setting additional bits in the upper region requires more than just the '_xor' command. Here are two examples that set the same value, one using the '_shl' command and one using the '_shr' command. To construct a value such as '0xf0000000', beginning with the 'ebx' register holding the value '0x80000000' and the 'ecx' register holding the value '0xffffffff', as above, the following can be used:

```
_push
_shr            ;ebx = ebx >> cl
_save           ;ecx = 1
_pop            ;ebx = 0x80000000 again
_push
_shr            ;ebx = 0x40000000
_push
_shr            ;ebx = 0x20000000
_push
_shr            ;ebx = 0x10000000
_save           ;ecx = 0x10000000
_pop
_xor            ;ebx = 0x30000000
_pop
_xor            ;ebx = 0x70000000
_pop
_xor            ;ebx = 0xf0000000
```

Whereas, to construct the value '0xf0000000', beginning with the 'ebx' and 'ecx' registers holding the value '0xffffffff', as above, the following can be used:

```
_push
_shr            ;ebx = ebx >> cl
_save           ;ecx = 1
_shl            ;ebx = 2
_xor            ;ebx = 3
_shl            ;ebx = 6
_xor            ;ebx = 7
_shl            ;ebx = 0x0e
_shl            ;ebx = 0x1c
_save           ;ecx = 0x1c
_pop
_shl            ;ebx = 0xf0000000
```

Thus, depending on the value, the '_shl' method is the simplest.

Astute readers will have noticed that none of the value constructions above use the '_addsaved' command. This shows that constants can be constructed without using any form of 'add'. However, it is also possible to perform the addition of arbitrary values without using any form of 'add', resulting in the removal of the '_addsaved' command by using this algorithm (edx and ebp holding the values to add together):

```
eax = edx ^ ebp
do
{
        ebp = (ebp & edx) << 1
        edx = eax
        eax = edx ^ ebp
}
while (edx & ebp)
ebx = eax
```

which we translate into this code:

```
[construct here the first value to add, not shown]
_nopdD      ;edx = ebx
[construct here the second value to add, not shown]
_nopdB      ;ebp = ebx
_save       ;ecx = ebx
_nopsD      ;ebx = edx
            ;optional, depending on the order of the
            ;constructions above
_xor        ;ebx = edx ^ ebp
_nopdA      ;eax = edx ^ ebp
_getEIP
_push       ;top of do-while loop
            ;ebx points to a hidden 'pop ebx'
            ;instruction as part of _getEIP so there
            ;is no explicit 'pop' instruction inside
            ;the loop that corresponds to this 'push'
            ;instruction
_saveJmpOff ;esi = ebx
_nopsD;ebx = edx
_save       ;ecx = edx
```

```
_nopsB      ;ebx = ebp
_and        ;ebx = ebp & edx
_push
_getDO      ;get data offset
_getdata    ;ebx = 0xffffffff
_save       ;ecx = 0xffffffff
_shr        ;ebx = 1
_save       ;ecx = 1
_pop
_shl        ;ebx = (edx & ebp) << 1
_nopdB      ;ebp = (edx & ebp) << 1
_save       ;ecx = ebx
_nopsA      ;ebx = eax
_nopdD      ;edx = eax
_push
_xor        ;ebx = edx ^ ebp
_nopdA      ;eax = edx ^ ebp
_pop
_and        ;ebx = edx & ebp
_JnzUp      ;loop while ((edx & ebp) != 0)
_pop        ;discard loop address
_nopsA      ;ebx = eax
```

30 (29) commands remain.

The replacement code for the '_addsaved' command requires the use of the base constant from the data section (and here, the value '1' would result in shorter code).

The value '1' can be constructed dynamically instead, in the following way:

```
_getEIP
_getdata        ;ebx=0xxxxxxx5b
_save           ;ecx=0xxxxxxx5b
_shl
_shr            ;ebx=0x1b
_save           ;ecx=0x1b
_addsaved       ;ebx=0x36
_addsaved       ;ebx=0x51
_addsaved       ;ebx=0x6c
_addsaved       ;ebx=0x87
_save           ;ecx=0x87
_shr            ;ebx=1
```

However, that algorithm prevents the removal of the '_addsaved' command. The two concepts seem to be mutually exclusive.

It is unclear whether the '_nopREAL' command could be removed, since there is no other single-byte command that might take its place in the event that a true 'no-operation' command were required. Its current purposes are to pad the unused slots following codon deletion and to fill the unused slot(s) that follow the '_JnzDown' command (since the '_JnzDown' command skips three slots). Note that the current implementation of the '_JnzDown' command contains a bug, which is that the destination of the branch is not the start of a slot. Instead, the command branches to two bytes past the start of the slot. The result is that the '_nopREAL' command must be used to fill that destination slot, otherwise a crash could occur because the branch might land in the middle of a command. However, the '_JnzDown' command can be removed by using alternative code, and any non-stack and non-memory instruction can be used for tail padding. Thus it appears that, given its current uses, the '_nopREAL' command can be removed.

29 (28) commands remain.

In the release version a '_null' command exists, which emits a single zero into the stream, followed by the 'nop' padding. Its existence is the result of a bug. The execution of such an instruction is likely to cause an exception. It is possible on *Windows XP* and later to register a vectored exception handle using the existing language, and that could intercept the exception, but this is quite outside the 'style' of the language. The command can be removed without any problem.

28 commands remain.

The '_JnzDown' command could be removed by using a careful implementation of '_JnzUp' (given that the meaning is reversed), but perhaps not without the loss of some functionality. It requires knowledge of the location of a forward branch destination. This interferes with command reordering if the buffer size is fixed, because there might not be enough slots available to construct the required 'add' value (unless the maximum number of slots was reserved each time in order to construct any possible number). It does, however, extend the functionality in a different way, since the '_JnzDown' command can skip only three commands at a time, requiring its use multiple times in order to execute larger conditional blocks. The '_JnzDown' command also places severe restrictions on what can appear in those conditional blocks, since an arithmetic operation might clear the Z flag, causing the branch to be taken instead of skipped. In contrast, the use of the '_JnzUp' command can skip an arbitrary number of commands without restriction. The difference can be demonstrated easily. We begin with some code that calls the GetTickCount() API to fetch a 'random' number (for ease of demonstration, the offset of the GetTickCount() API is set arbitrarily to the value '0x0c'), using the '_JnzDown' command:

```
;construct pointer to GetTickCount()
;construct the value "0x0c"

_getDO       ;get data offset
_getdata     ;ebx = 0xffffffff
_save        ;ecx = 0xffffffff
_shr         ;ebx = 1
_save        ;ecx = 1
_shl         ;ebx = 2
_xor         ;ebx = 3
```

```
_shl          ;ebx = 6
_shl          ;ebx = 0x0c
_nopdB        ;ebp = 0x0c
_save         ;ecx = 0x0c

;add to data offset

_getDO        ;get data offset
_nopdD        ;edx = data offset
_xor          ;ebx = edx ^ ebp
_nopdA        ;eax = edx ^ ebp
_getEIP
_push         ;top of do-while loop
              ;ebx points to a hidden 'pop ebx'
              ;instruction as part of _getEIP
              ;so there is no explicit 'pop'
              ;instruction inside the loop
              ;that corresponds to this 'push'
              ;instruction
_saveJmpOff   ;esi = ebx
_nopsD        ;ebx = edx
_save         ;ecx = edx
_nopsB        ;ebx = ebp
_and          ;ebx = ebp & edx
_push
_getDO        ;get data offset
_getdata      ;ebx = 0xffffffff
_save         ;ecx = 0xffffffff
_shr          ;ebx = 1
_save         ;ecx = 1
_pop
_shl          ;ebx = (edx & ebp) << 1
_nopdB        ;ebp = (edx & ebp) << 1
_save         ;ecx = ebx
_nopsA        ;ebx = eax
_nopdD        ;edx = eax
_push
_xor          ;ebx = edx ^ ebp
_nopdA        ;eax = edx ^ ebp
_pop
_and          ;ebx = edx & ebp
_JnzUp        ;loop while ((edx & ebp) != 0)
_pop          ;discard loop address
_nopsA        ;ebx = eax

;call GetTickCount()

_call
```

Then the choice is made, and the branch might be taken (seven in eight chances to take it):

```
;construct the value '7'

_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_shr            ;ebx = 1
_save           ;ecx = 1
_shl            ;ebx = 2
_xor            ;ebx = 3
_shl            ;ebx = 6
_xor            ;ebx = 7
_save           ;ecx = 7

;'and' with result from GetTickCount()

_nopsA
_and            ;ebx = ebx & 7
_JnzDown
[conditional command 1]
[conditional command 2]
[conditional command 3]
_nopREAL        ;work around '_JnzDown' bug
```

The replacement code might look something like this, beginning immediately after the call to the GetTickCount() API:

```
    ;save result from GetTickCount()

    _nopsA
    _push

    ;construct pointer to l2

    _getDO       ;get data offset
    _getdata     ;ebx = 0xffffffff
    _save        ;ecx = 0xffffffff
    _shr         ;ebx = 1
    _save        ;ecx = 1
    ... ['_shl' and '_xor' as needed to produce the
    value ((lines(l1...l2) * 8) + 3)]
    _nopdB       ;ebp = offset of l2
    _save        ;ecx = offset of l2
    _getEIP
l1: _nopdD       ;edx = eip
    _xor         ;ebx = edx ^ ebp
    _nopdA       ;eax = edx ^ ebp
    _getEIP
    _push        ;top of do-while loop
                 ;ebx points to a hidden 'pop ebx'
                 ;instruction as part of _getEIP
                 ;so there is no explicit 'pop'
                 ;instruction inside the loop
                 ;that corresponds to this 'push'
                 ;instruction
    _saveJmpOff  ;esi = ebx
    _nopsD       ;ebx = edx
    _save        ;ecx = edx
    _nopsB       ;ebx = ebp
    _and         ;ebx = ebp & edx
    _push
    _getDO       ;get data offset
    _getdata     ;ebx = 0xffffffff
    _save        ;ecx = 0xffffffff
    _shr         ;ebx = 1
    _save        ;ecx = 1
    _pop
    _shl         ;ebx = (edx & ebp) << 1
    _nopdB       ;ebp = (edx & ebp) << 1
    _save        ;ecx = ebx
    _nopsA       ;ebx = eax
```

```
    _nopdD      ;edx = eax
    _push
    _xor        ;ebx = edx ^ ebp
    _nopdA      ;eax = edx ^ ebp
    _pop
    _and        ;ebx = edx & ebp
    _JnzUp      ;loop while ((edx & ebp) != 0)
    _pop        ;discard loop address
    _nopsA      ;ebx = eax
    _saveJmpOff

    ;restore result from GetTickCount()

    _pop
    _nopdA      ;eax = GetTickCount()

    ;construct the value '7'

    _getDO      ;get data offset
    _getdata    ;ebx = 0xffffffff
    _save       ;ecx = 0xffffffff
    _shr        ;ebx = 1
    _save       ;ecx = 1
    _shl        ;ebx = 2
    _xor        ;ebx = 3
    _shl        ;ebx = 6
    _xor        ;ebx = 7
    _save       ;ecx = 7

    ;'and' with result from GetTickCount()

    _nopsA      ;ebx = GetTickCount()
    _and        ;ebx = ebx & 7
    _JnzUp
    [conditional command 1]
    [conditional command 2]
    [conditional command 3]
    ...
    [conditional command n]
l2:     ;branch destination is here
```

27 commands remain.

In the same way as for the '_JnzDown' command, the '_JzDown' command can be removed.

26 commands remain.

Normally, the 'ecx', 'esi' and 'edi' registers are write-only (technically, the 'ecx' register only becomes write-only after the '_addsaved' command is removed), leaving the 'eax', 'ebx', 'edx' and 'ebp' registers as general-purpose registers. However, there is a way to read these registers again after they have been written. The '_pushall' command pushes the registers onto the stack in this order: eax, ecx, edx, ebx, esp, ebp, esi, edi. The registers can then be popped individually from the stack, by using the '_pop' command, in the following way:

```
_pushall   ;save all registers
_pop       ;edi
_pop       ;esi
_pop       ;ebp
_pop       ;esp (useful for reading stack parameters,
           ;using the '_getdata' command, see below)
_pop       ;ebx
_pop       ;edx
_pop       ;ecx
_pop       ;eax
```

A smaller set of '_pop' commands can be used to access particular registers, leaving the others for removal later, if necessary. The popped registers can also be modified and pushed back onto the stack, allowing the '_popall' command to be used to pop all of them. This allows multiple values to be assigned simultaneously. By combining several of these tricks, it becomes possible to remove the '_mul' command (edx:eax = eax * ebx). A working solution can be downloaded from http://pferrie.tripod.com/misc/flibi_mul.zip.

25 commands remain.

Interestingly, by reordering the register initialization code for the first addition block to remove one instruction, the code actually increases in size because the branch to l4 requires more instructions to construct it as a result. This brings us to a special-case problem of dynamic pointer construction. There is a particular problem when the code at l2 branches to l4 and the code at l3 branches to l1, but where l1 < l2 and l4 > l3, as shown here:

```
l1:    [code]
l2:    jz      l4
l3:    jnz     l1
l4:    [code]
```

First, construct the branch from l2 to l4:

```
l1:    [code]

       ;construct relative to l2 (two instructions)

       mov     reg, 1
       shl     reg, 1
l2:    jz      l2+reg
l3:    jnz     l1
l4:
[code]
```

Then construct the branch from l3 to l1:

```
l1:    [code]
       mov     reg, 1
       shl     reg, 1
l2:    jz      l2+reg

       ;construct relative to l3 (four lines)
       ;[code] at l1 is a single instruction to keep
       ;the example simple

l3:    mov     reg, 1
       shl     reg, 1
       shl     reg, 1
       jnz     l3-reg
l4:    [code]
```

Now the branch at l2 is affected, and no longer points to l4, so reconstruct it:

```
l1:     [code]

        ;construct relative to l2 (five instructions)

        mov     reg, 1
        shl     reg, 1
        shl     reg, 1
        add     reg, 1
l2:     jz      l2+reg
l3:     mov     reg, 1
        shl     reg, 1
        shl     reg, 1
        jnz     l3-reg
l4:     [code]
```

But now the branch at l3 is affected, and no longer points to l1, so reconstruct it:

```
l1:     [code]
        mov     reg, 1
        shl     reg, 1
        shl     reg, 1
        add     reg, 1
l2:     jz      l2+reg

        ;construct relative to l3 (six instructions)

l3:     mov     reg, 1
        shl     reg, 1
        add     reg, 1
        shl     reg, 1
        jnz     l3-reg
l4:     [code]
```

Again, the branch at l2 is affected and no longer points to l4, so reconstruct it:

```
l1:     [code]

        ;construct relative to l3 (six instructions)

        mov     reg, 1
        shl     reg, 1
        add     reg, 1
        shl     reg, 1
l2:     jz      l2+reg
l3:     mov     reg, 1
        shl     reg, 1
        add     reg, 1
        shl     reg, 1
        jnz     l3-reg
l4:     [code]
```

Finally, the instructions are reordered but not inserted, and the combination works. The limitation is that the lines in the construction must converge on a multiple of each other. Such a value might not exist without the explicit insertion of 'alignment' lines. The '_nop' command could be used for this purpose, but any 'harmless' instruction can be used, such as moving to/from the same register from/to which a value was just moved (more specifically, if the previous move instruction was from ebx to eax, then it is harmless to move from eax back into ebx). By combining several of these tricks, it becomes possible to remove the '_div' command (eax, edx = edx:eax / ebx) as well. A working solution can be downloaded from http://pferrie.tripod.com/misc/flibi_div.zip.

24 commands remain.

The '_writeDWord' command can be removed by using this algorithm:

```
mov     [edi], bl
inc     edi
shr     ebx, 8
mov     [edi], bl
inc     edi
shr     ebx, 8
mov     [edi], bl
inc     edi
shr     ebx, 8
mov     [edi], bl
```

which we translate into this code:

```
;construct the value '8'

_push
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_shr            ;ebx = 1
_save           ;ecx = 1
_shl            ;ebx = 2
_shl            ;ebx = 4
_shl            ;ebx = 8

;save in ecx for later

_save           ;ecx = 8
_pop

;write byte 0

_writeByte

;increment edi

_pushall
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_shr            ;ebx = 1
_nopdB          ;ebp = 1
_save           ;ecx = 1
_pop            ;ebx = edi
_nopdD          ;edx = edi
_xor            ;ebx = edx ^ ebp
_nopdA          ;eax = edx ^ ebp
_getEIP
_push   ;top of do-while loop
        ;ebx points to a hidden 'pop ebx' instruction
        ;as part of _getEIP so there is no explicit
        ;'pop' instruction inside the loop that
        ;corresponds to this 'push' instruction
```

```
_saveJmpOff     ;esi = ebx
_nopsD          ;ebx = edx
_save           ;ecx = edx
_nopsB          ;ebx = ebp
_and            ;ebx = ebp & edx
_push
_getDO          ;get data offset
_getdata        ;ebx = 0xffffffff
_save           ;ecx = 0xffffffff
_shr            ;ebx = 1
_save           ;ecx = 1
_pop
_shl            ;ebx = (edx & ebp) << 1
_nopdB          ;ebp = (edx & ebp) << 1
_save           ;ecx = ebx
_nopsA          ;ebx = eax
_nopdD          ;edx = eax
_push
_xor            ;ebx = edx ^ ebp
_nopdA          ;eax = edx ^ ebp
_pop
_and            ;ebx = edx & ebp
_JnzUp          ;loop while ((edx & ebp) != 0)
_pop            ;discard loop address
_nopsA          ;ebx = eax

;update edi

_push
_popall         ;edi = eax and rebalance stack

;shift ebx right by 8

_shr            ;ebx = ebx >> 8

;write byte 1

_writeByte

[repeat twice more, beginning with 'increment edi'
from above, to write the remaining bytes]
```

Of course, if there were a command to write a new value for the stack pointer, then the stack could be moved to the destination address, and a '_push' command could be used to write the value. However, there would need to be a corresponding command to read the previous value for the stack pointer in order to restore it afterwards. This is quite outside the 'style' of the language.

23 commands remain.

Another instruction that can be removed is the '_call' command. A subroutine call is equivalent to pushing the return address onto the stack, and then jumping to the location of the subroutine. It can be replaced by the '_JnzUp' command in the following way (again, calling the GetTickCount() API, as above):

```
    ;construct pointer to l2

    _getDO      ;get data offset
    _getdata    ;ebx = 0xffffffff
    _save       ;ecx = 0xffffffff
    _shr        ;ebx = 1
    _save       ;ecx = 1
    ... ['_shl' and '_xor' as needed to produce the
    value ((lines(l1...l2) * 8) + 3)]
    _nopdB      ;ebp = offset of l2
    _save       ;ecx = offset of l2
    _getEIP
l1: _nopdD      ;edx = eip
    _xor        ;ebx = edx ^ ebp
    _nopdA      ;eax = edx ^ ebp
    _getEIP
    _push       ;top of do-while loop
        ;ebx points to a hidden 'pop ebx' instruction
        ;as part of _getEIP so there is no explicit
        ;'pop' instruction inside the loop that
        ;corresponds to this 'push' instruction
    _saveJmpOff ;esi = ebx
    _nopsD      ;ebx = edx
    _save       ;ecx = edx
    _nopsB      ;ebx = ebp
    _and        ;ebx = ebp & edx
    _push
    _getDO      ;get data offset
    _getdata    ;ebx = 0xffffffff
    _save       ;ecx = 0xffffffff
    _shr        ;ebx = 1
    _save       ;ecx = 1
    _pop
    _shl        ;ebx = (edx & ebp) << 1
    _nopdB      ;ebp = (edx & ebp) << 1
    _save       ;ecx = ebx
    _nopsA      ;ebx = eax
    _nopdD      ;edx = eax
    _push
    _xor        ;ebx = edx ^ ebp
    _nopdA      ;eax = edx ^ ebp
    _pop
    _and        ;ebx = edx & ebp
    _JnzUp      ;loop while ((edx & ebp) != 0)
    _pop        ;discard loop address
    _nopsA      ;ebx = eax

    ;save return address on stack

    _push

    ;construct pointer to GetTickCount()

    _getDO      ;get data offset
    _getdata    ;ebx = 0xffffffff
    _save       ;ecx = 0xffffffff
    _shr        ;ebx = 1
    _save       ;ecx = 1
    _shl        ;ebx = 2
    _xor        ;ebx = 3
    _shl        ;ebx = 6
    _shl        ;ebx = 0x0c
    _nopdB      ;ebp = 0x0c
    _save       ;ecx = 0x0c
    _getDO      ;get data offset
```

```
    _nopdD       ;edx = data offset
    _xor         ;ebx = edx ^ ebp
    _nopdA       ;eax = edx ^ ebp
    _getEIP
    _push      ;top of do-while loop
               ;ebx points to a hidden 'pop ebx'
               ;instruction as part of _getEIP so
               ;there is no explicit 'pop'
               ;instruction inside the loop that
               ;corresponds to this 'push' instruction
    _saveJmpOff  ;esi = ebx
    _nopsD       ;ebx = edx
    _save        ;ecx = edx
    _nopsB       ;ebx = ebp
    _and         ;ebx = ebp & edx
    _push
    _getDO       ;get data offset
    _getdata     ;ebx = 0xffffffff
    _save        ;ecx = 0xffffffff
    _shr         ;ebx = 1
    _save        ;ecx = 1
    _pop
    _shl         ;ebx = (edx & ebp) << 1
    _nopdB       ;ebp = (edx & ebp) << 1
    _save        ;ecx = ebx
    _nopsA       ;ebx = eax
    _nopdD       ;edx = eax
    _push
    _xor         ;ebx = edx ^ ebp
    _nopdA       ;eax = edx ^ ebp
    _pop
    _and         ;ebx = edx & ebp
    _JnzUp       ;loop while ((edx & ebp) != 0)
    _pop         ;discard loop address
    _nopsA       ;ebx = eax
    _getdata     ;ebx = offset of GetTickCount()
    _saveJmpOff  ;esi = offset of GetTickCount()

    ;clear Z flag

    _save        ;ecx = ebx
    _and         ;ebx = ebx & ebx (known non-zero from
                 ;above)

    ;jump to GetTickCount()

    _JnzUp
12: ;execution resumes here
```

Local subroutines can be called in the same way; however there is no 'return' command. The equivalent for a 'return' command is the following:

```
;retrieve return address from stack

_pop
_saveJmpOff     ;esi = return address

;clear Z flag, if required

_save   ;ecx = ebx
_and    ;ebx = ebx & ebx (known non-zero from above)

;return to caller

_JnzUp
```

22 commands remain.

The following are two useful tricks just for the sake of interest. The first one demonstrates how to read parameters directly from the stack:

```
[push parameters here, not shown]
_pushall
_pop            ;edi
[_nopdA         ;eax = edi, if needed]
_pop            ;esi
[_nopdD         ;edx = esi, if needed]
_pop            ;ebp (discard)
_pop            ;esp
_push           ;esp
_push           ;ebp
[_nopsD         ;ebx = original esi, if needed]
_push
[_nopsA         ;ebx = original edi, if needed]
_push
_popall         ;ebp = esp
[add to ebp as needed to reach required variable]
_nopsB          ;ebx = ebp
_getdata        ;read from stack
```

Then, simply by replacing the '_getdata' command with the '_call' command, function pointers on the stack can be called.

The '_push' command can be removed, but the replacement code is ugly. It would look like this:

```
_nopdA    ;place into eax in order to appear at the
          ;top of the stack
_pushall
_pop            ;discard edi
_pop            ;discard esi
_pop            ;discard ebp
_pop            ;discard esp
_pop            ;discard ebx
_pop            ;discard edx
_pop            ;discard ecx
;eax remains as the only register on the stack
```

21 commands remain.

The '_popall' command can be removed. The '_popall' command pops the registers from the stack in the following order: edi, esi, ebp, esp, ebx, edx, ecx, eax. The command can be replaced by popping and assigning the registers individually, in the following way:

```
_pop            ;edi
_saveWrtOff
_pop            ;esi
_saveJmpOff
_pop            ;ebp
```

```
_nopdB
_pop            ;esp (discard)
_pop            ;ebx
_pop            ;edx
_nopdD
_pop            ;ecx
_save
_pop            ;eax
_nopdA
```

20 commands remain.

The '_nopdB', '_saveWrtOff' and '_saveJmpOff' commands can be removed if the '_push' and '_popall' commands are retained. Replacement of the '_saveWrtOff' command would look like this:

```
_pushall
_pop            ;discard existing edi
[construct value to place into edi, not shown]
_push
_popall
```

Replacement of the '_saveJmpOff' command would look like this:

```
_pushall
_pop            ;edi
[_nopdD         ;preserve edi if needed]
_pop            ;discard existing esi
[construct value to place into esi, not shown]
_push
[_nopsD         ;restore edi if needed]
_push
_popall
```

Replacement of the '_nopdB' command would look like this:

```
_pushall
_pop            ;edi
[_nopdD         ;preserve edi if needed]
_pop            ;esi
[_nopdA         ;preserve esi if needed]
_pop            ;discard existing ebp
[construct value to place into ebp, not shown]
_push
[_nopsA         ;restore esi if needed]
_push
[_nopsD         ;restore edi if needed]
_push
_popall
```

19 commands remain.

Two other commands can be removed, but they cannot be replaced using existing instructions. Instead, the replacement code requires the introduction of another instruction. The two commands are '_shl' and '_shr'. The replacement instruction is '_rot' ('rotate'). The direction of the rotate is not important, as long as it is known, since all values can be constructed by using it in conjunction with the '_and' instruction. However, it requires the use of the value '1' as the 'base constant'. The value '1' would be used to construct the values '0x7fffffff' (if rotating shifts to the right) or '0xfffffffe' (if rotating shifts to the left). This is the mask value that is used by the '_and' command to zero the appropriate bit in order to simulate a shift. This is the simplest implementation that would rotate a value only once per use without reference to the value in the 'ecx' register. Multi-bit rotates could be supported, too, but then the 'and' mask would no longer be a constant. Instead, it would be specific to the number of bits that are being rotated. So, shifting the value in the 'eax' register left by '3' times, using the single-bit rotate command, would look like this:

```
;construct the value '0xfffffffe'

_getDO          ;get data offset
_getdata        ;ebx = 1
_save           ;ecx = 1
_rot            ;ebx = 2
_xor            ;ebx = 3
_rot            ;ebx = 6
_xor            ;ebx = 7
_rot            ;ebx = 0x0e
_xor            ;ebx = 0x0f
[repeat seven more times, but omit the final xor]
_save           ;ecx = 0xfffffffe

;rotate left and zero the overflow bits

_nopsA          ;ebx = eax
_rot            ;ebx = rol(ebx, 1)
_and            ;ebx = ebx & 0xfffffffe
_rot            ;ebx = rol(ebx, 1)
_and            ;ebx = ebx & 0xfffffffe
_rot            ;ebx = rol(ebx, 1)
_and            ;ebx = ebx & 0xfffffffe
_nopdA          ;eax = shl(eax, 3)
```

18 commands remain:

- _nopsA, _nopsB, _nopsD, _nopdA, _nopdD
- _writeByte
- _save
- _getDO, _getdata, _getEIP
- _push _pop, _pushall, _popall
- _rot, _and, _xor
- _JnzUp

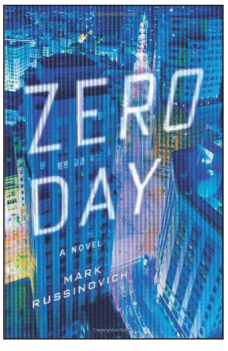
Many years from now, our distant descendants might stumble upon a codon stream that describes only 18 amino acids – and we might be looking at its origin. Imagine that.

BOOK REVIEW

ZERO DAY: A NOVEL

Paul Baccas
Sophos, UK

VB usually reserves book reviews for factual books dedicated to the subject of information security. This month, however, we break away from tradition to review a piece of fiction written by renowned Windows systems internals expert and Microsoft Technical Fellow, Mark Russinovich.



Title: Zero Day

Author: Mark E. Russinovich

Publisher: Saint Martin's Press Inc. (4 Jan 2011)

Pages: 336 (hardcover)

ISBN-13: 978-0312612467

RRP: £17.99 (hardcover)

Before I start, let me say that I am an omni-lector (reader of all), and while thrillers are not usually my genre of choice I do, on occasion, enjoy them. When *Virus Bulletin* asked me to review a thriller I was happy to oblige and awaited its arrival with a mixture of excitement and apprehension.

The arrival of the novel coincided with a few days of unusually warm spring weather and I was afforded the rare luxury of some outdoor reading time while I got to grips with the plot.

SYNOPSIS

The book's main character, Jeff Aiken, is an independent security researcher who is scarred from time spent working for the US government. He is called to New York City – somewhere he hasn't visited since his girlfriend died in the 9/11 attacks on the Twin Towers – to investigate a computer system failure. Aiken is racked with guilt because, in the weeks leading up to the 9/11 attacks, he had found evidence to suggest that such a terrorist attack was likely. As he begins his investigation of the computer failures in New York a disturbing series of problems on other critical systems starts to unravel and Aiken fears another attack.

SUPPORTERS

The dust jacket boasts comments from some pretty impressive names: the authors Nelson DeMille and William Landry; White House Cyber Security Coordinator Prof. Howard A. Schmidt (who has also written a foreword); and the entrepreneur and philanthropist Bill Gates all sing the book's praises.

IS THE STORY TECHNICALLY BELIEVABLE?

There are long and short answers to this question. The short answer is yes – the writing makes enough sense for the errors/misapprehensions about malware and anti-malware techniques to be lost in the flow of the story. The long answer is that, while Mark is an expert in *Windows* systems and rootkits, he isn't an expert on the anti-malware industry, and vendors are portrayed in a very naïve way. If we ignore the premise that vendors are bad and the government is good at fighting malware, the rest of the book is technically believable (although one also hopes that nuclear power stations aren't running *Windows* in the real world).

The book is divided into five sections corresponding to four weeks' build-up and the aftermath. The first half of the novel reminds me of some of Michael Crichton's stories – particularly *Airframe* – and as a whole the novel is very filmic. It is very teachy, though, and explaining that 'the kingdom' is how Saudis refer to their country since the 2007 movie of the same name put the term into common parlance is a little *too* teachy.

The second half of the novel moves into action after the cerebral beginnings and at that point the plot begins to lose a little of its integrity. An editor should have tightened this up and a screen writer would have to.

VERDICT

I suspect that the book will make it to the big screen as it has all the elements of a movie: a dashing hero and beautiful heroine (which security conferences has Mark been attending?) with a fast-paced story line that screams 'film me'. It even has the customary bad guy with an English accent.

I believe that the three elements of a genre novel are plot, characterization and idea. Scoring these out of five I would give *Zero Day*:

- Plot: 3–4
- Characterization: 3
- Idea: 4

The main characters are well formed, but others are slightly more one-dimensional. The idea is good and the plot fast-paced. I would buy this book, and if you are looking for some holiday reading then you could do a lot worse than getting your hands on a copy.

PRODUCT REVIEW

AGNITUM OUTPOST SECURITY SUITE FREE

John Hawes

Free anti-virus continues to be all the rage, with more and more firms joining in the free-for-all (or at least, free for all non-commercial purposes). The days when the choice of free solutions was limited to the big As, *Avast* and *AVG*, are long gone, and the open market is now crowded with competing solutions. The business model is generally based on persuading home users that a product is so good it's worth using at work too – or else hooking people on a basic model and getting them to upgrade to a more complete suite. Depending on who you ask, this approach is either seen to be a great way of reaching out to a wider audience and generating interest in a brand, or as a desperate scrabble for space in a field full to bursting with highly competitive solutions.

The latest company to offer its protection free of charge is *Agnitum*, originally best known for its highly regarded firewall solution (which has been available as a free offering for a while), and now also producing a solid and fairly complete suite solution which has built up a very decent reputation in our regular tests. While in many cases free offerings are pared-down, AV-only solutions, *Agnitum* is making the full suite available free to home users, with just a few modifications and the usual provisos about commercial use etc. We took the product into the lab to have a look at the user experience and see just how much you can get for nothing these days.

COMPANY AND WEB PRESENCE

Agnitum was set up in 1999, so has a fair bit of history behind it. The St Petersburg-based company launched the *Outpost* firewall brand in 2002, and expanded into anti-spam in 2007, with the full suite product – integrating anti-virus detection courtesy of the ever-popular *VirusBuster* engine – emerging later the same year. Throughout this period at least some part of the company's product line seems to have been given away free.

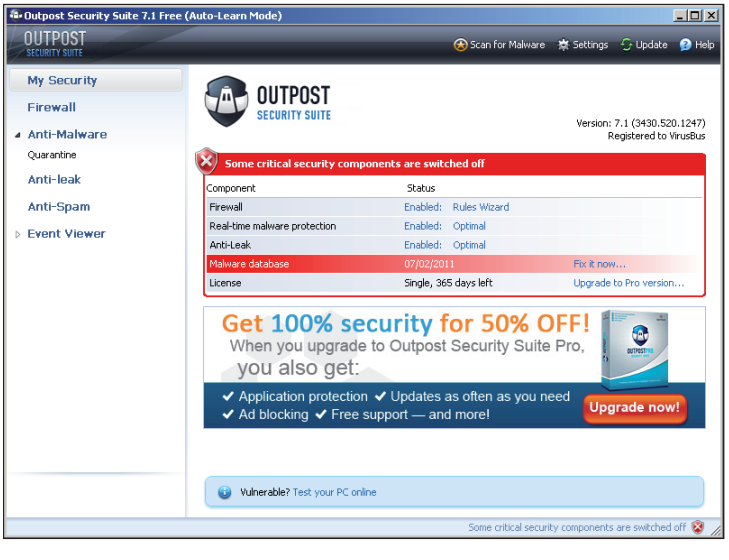
The company's website has recently had a bit of a face lift and looks clean and efficient without overdoing the glitter and glitz. The 'About' section boasts an impressive list of technology partners making use of *Agnitum*'s developments (mainly the firewall); the roster includes the likes of *AVG*, *BullGuard*, *Lavasoft*, *Novell*, *Quick Heal* and *Sophos*.

The rest of the website is fairly unexceptional; the home page features sizeable advertisements for the company's headline products (gratifyingly adorned with the VB100 logo the company has earned fairly reliably for the last few years). The bulk of the site is given over to product-related content, with sales and downloads taking centre stage; the company has a number of innovative licensing deals, including multi-system, multi-year deals, and even lifetime offerings.

Unlike many rival firms seeking to combine education with news presence, little attention is paid to information on specific threats or threat-related news stories. The 'news' and blog sections focus almost exclusively on upcoming release schedules, product features, awards and so on – the awards page features a lengthy roster of badges and accolades from a variety of download sites, magazine reviews and testing organizations, with high scores from firewall leak tester *Matousec* prominent throughout.

Of most interest for the majority of users (those who are not die-hard *Outpost* fans that is), will be the support area. The landing page of this section leads in with a good list of major FAQs, all answered lucidly and in ample depth. This is backed up by an even more extensive knowledgebase section, again with each question covered with impressive detail and clarity. Our only quibble would be that we would prefer to see the newest and most widely used products listed at the top of the dropdown list, rather than at the bottom.

The documentation section is properly sorted, and unlike many security vendor sites where manuals are hidden away like some embarrassing aged relative, here they are given a prominent position and made easy to find and access. A quick skim through some of these showed them to be well designed, clear and thorough. *Agnitum* should be congratulated for paying the right degree of attention to its documentation, clearly realising the importance



of complete, detailed and usable information about its products.

INSTALLATION AND CONFIGURATION

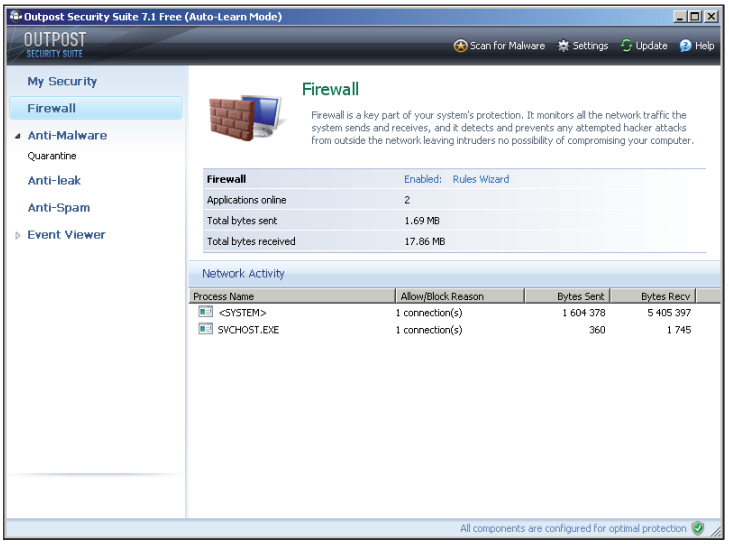
The free version of *Agnitum*'s suite is not as easy to get hold of as one might expect, with minimal reference made to it on the company's website – a link can be found quietly displayed halfway down a list on the 'Products' page. The link leads to a mini-site at free.agnitum.com, and from here, download links lead to *CNET*'s download.com. The mini-site is fairly rudimentary, but does host a nice screenshot of the product, and some information on awards received and comparison with other free solutions. The company boasts that this is the first fully featured Internet security suite to be given away in this manner – its closest rival, from *Comodo*, lacking the anti-spam feature which is a fairly standard component of any full suite. The site also makes clear what we are missing out on by not going for the full 'Pro' version of the product – including the fact that the free version features only limited support for multiple languages, less than complete 'Safe web surfing', an absent 'Unique ID protection' module, and no 'priority updates'. Both 32-bit and 64-bit versions are available, and supported systems include *XP*, *Vista* and *Windows 7* with a minimum of 256MB of RAM and 400MB of disk space.

The downloaded package is an executable of 85–90MB, which runs through the installation process in pretty standard manner, starting as always with the choice of languages. The selection of languages is limited (as the website warned) to English or German. The opening salvo of the installer proper offers the option to upgrade to the full 'Pro' product (offering a free 30-day trial), and once again displays a table showing the differences between the two. This time there are more areas in which the free edition is shown to be inferior, with the fact that the product cannot be used in a corporate environment added to the mix. A EULA comes next, and includes a warning that the company's support staff may not respond to problems reported by non-paying users (they will do so only if they have time), and there is an option to join a community feedback system labelled 'ImproveNet'. The installation process itself then commences, completing in just a minute or two, including setting up network filters and running a 'smart scan' to collect information on the machine and its environment. A reboot is required to round things off, which is fairly standard in products offering such in-depth protection, although steadily going out of favour with anti-malware only solutions.

On restart, it was something of a surprise to find a prompt insisting that the product be registered. Once again, we were reminded of the benefits of upgrading to the paid version, and told that the free edition must be registered online (or by phone or email) within two days. This process is fairly simple, requiring only a username and email address, to which a (lengthy) activation code key is sent. With this all done, we finally got a look at the interface itself.

The interface was pretty similar to the 'Pro' product we have looked at in many VB100 tests in recent years, and which we have always found to be clearly laid out and fairly simple to operate. The most striking difference is a sizeable advert dominating the bottom half of the GUI, promoting a discounted version of the 'Pro' edition. It also warns, by circling the main information area in red and highlighting one of the entries, that complete security is not applied. The entry in question refers to the updates, which report being several weeks old, and an accompanying button offers to 'fix it now'. Rather than initiating an update as one might expect, though, the result of clicking this button is that a web page is opened which once again promotes the value of the paid-for edition over the free one, referring to the possible inadequacy of the free version's 'no-guarantee update schedule'. Something of a pattern was beginning to emerge.

We quickly found that updating itself can easily be initiated using the update button on the toolbar at the top of the interface, and this time no complaints were made about using the free version. The update itself took around five minutes the first time it was run; later retries were much quicker. As the company repeatedly points out, relying on the user to remember to update is less than ideal, with most full products updating automatically multiple times per day at least, and some performing tiny updates every few minutes, or even relying on online databases to protect from the latest threats. It seems like quite some window of vulnerability may well be left open here.



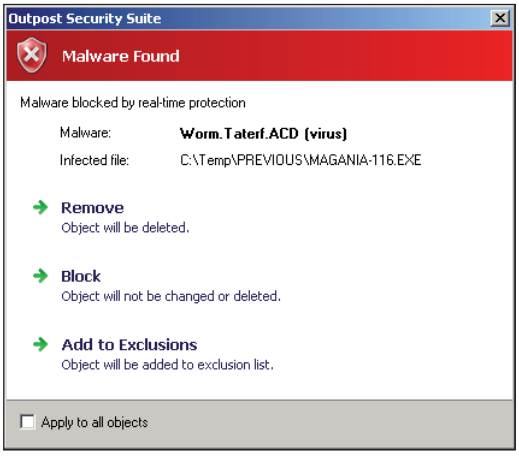
The remainder of the product is much like the full version. The firewall section, which is *Agnitum*'s speciality, is given pride of place, and the control system is admirably clear and simple to operate. It does require some degree of understanding to operate properly in-depth, but seems accessible to most users at its basic level. A simple slider allows a selection of standard approaches, with the default being to offer a rule-creation wizard each time an unknown process or activity is observed. There is perhaps a little less explanation built in than we have seen in some products, but the simplicity is aided by a lack of clutter and wordiness, and many sections are accompanied by links to the online knowledgebase, where clear and detailed explanations of some of the more complex areas of firewall configuration are provided.

The controls for the firewall are actually part of a generic settings section, also accessible via a link on the main toolbar and from various other places. Each section here is given similar treatment, with simplicity and clarity the order of the day. The anti-malware section provides a decent level of configuration, without going into the depth that some products offer. Coverage of the standard areas, such as areas to scan and exclusions, limits to scanning and actions on detection, are provided for the various types of detection on offer, including the real-time and web scanners as well as a selection of on-demand scan types.

Also included in the controls are sections covering application control and leak prevention. Along with the firewall, these interact and overlap to form a complete set of rules for what applications can do across the network, along with some basic control of what can be performed locally. The anti-spam controls also have their own section – which, again, is laid out simply and clearly – and finally some logging controls are provided, covering both the size of logs and the type of data that is recorded.

MALWARE DETECTION AND PROTECTION

The main anti-malware part of the product is based on the *VirusBuster* engine. The control system, described above, is very well designed and simple to operate, and as we have found in many comparative tests in recent years, the product is well implemented and runs with great stability and ruggedness under extreme pressure. Putting the free edition through some extra heavy tests showed it to be just as resilient, with none of the system slowdowns, GUI freak-outs or other bad behaviours we have seen recently in many lesser products. We have also been impressed with the speed and resource consumption of the *Agnitum* products we have tested in comparisons lately, with the caching of previous activities making for lightning fast speeds and minimal overheads once the product has settled into its

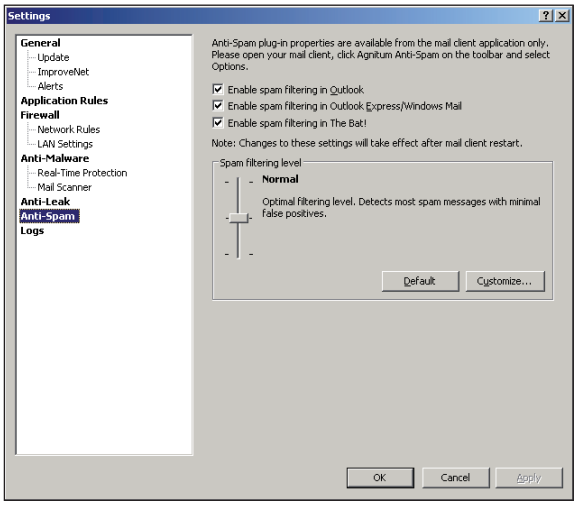


environment; the free edition showed itself to be similarly efficient and smartly designed.

Detection rates for the *VirusBuster* engine have been increasing steadily over recent years. While not quite up there with the leaders in terms of pure static detection rates, the engine is clearly solid and reliable, with a dependable record of VB100 passes too. Running the product through some additional tests, figures seemed comparable with our expectations. With the default settings, detection of malware brings up a very simple dialog offering to remove or simply block the offending item. Somewhat oddly, in some cases we saw detected items prevented from writing to disk, while others were warned about but written happily; those which were written could not then be moved elsewhere or executed, so they remained fairly safe.

When those items which were not spotted by the standard detection mechanism were allowed to execute on a system, we saw some good protection provided by the application control and leak prevention systems – either preventing changes to core system components or stopping items from connecting outwards to perform malicious activities. In some cases items were allowed to copy themselves to system folders, set themselves to auto-start on boot etc., but for the most part at least some action was taken to mitigate the impact of an infection.

The anti-leak system has been rigorously analysed by other specialist labs and awarded repeated high ratings, and in our brief and unscientific look at its reliability we saw no reason to disagree with these conclusions. We hope to initiate some more formal testing of dynamic protective measures in the near future, and it will be interesting to see how the limited updating of the standard anti-malware component balances out with this solution's higher level of additional protection compared to most other free offerings.



OTHER FEATURES

We've already had a quick look at the control systems for the various components, and briefly mentioned some of the capabilities of the application control and anti-leak layers. The main area remaining is the anti-spam component – the portion which marks this suite out from other freebies. This supports *Outlook*, *Outlook Express* (*Windows Mail* in newer versions) and *The Bat!*, a popular mail client in Russia and eastern Europe.

The controls here are fairly basic, with few difficult technical questions being asked. The user has the option to set detection levels to high, standard or low, or can fine-tune the exact levels at which messages will be adjudged to be spam or suspicious. An additional set of controls, accessible from within the mail client, allow tweaking of blacklists and whitelists, and the marking and removal of items thought to be spam. Existing classified mail can also be scanned and added to the self-training data used to adjust the detection to the needs of the specific user. Our anti-spam testing system is geared mainly towards server-level solutions, but we hope to be able to measure the performance of desktop products in the near future, and to take a closer look at how well the anti-spam component of this suite operates.

That covers most of what the suite has to offer. As a free solution, we cannot really complain at the absence of any additional components above and beyond the standard set of items expected in a suite. While some paid-for products may include system monitoring tools, parental controls and other odds and ends, this product focuses on covering the standard bases of anti-malware, spam filtering, firewalling and application control.

The only remaining section of the product interface is the event viewer, which corresponds roughly to the 'Logs' section of the configuration system, and here the user can monitor all events recorded by the firewall, anti-malware and anti-leak components in real time, as well as viewing some information about the product in general. This is as clear and pleasant as any such system, and may repay some attention from more scrupulous users.

CONCLUSIONS

Overall, it is pretty hard to find fault with this product – especially as a free solution. It is almost identical in most respects to the full 'Pro' edition, which we have been testing for a long time. The main difference is in the updating, which is somewhat less effective for being performed once a day rather than more regularly. This approach to free solutions is not entirely unusual though, with some making it clear that users of the free product are given a lower priority than paying users when updates are provided. Many paid-for products also default to daily updates unless specifically set to be more rigorous by the user.

In this case the vendor has taken the idea of reminding the user that paid-for versions are superior to the free one somewhat to extremes – repeatedly insisting that users are putting themselves at risk if they stick to an incomplete and inferior product. Only time will tell whether this 'nagging' will result in the user upgrading to the full product (as intended), or whether the user will instead be so irritated that they resort to removing the product and implementing an alternative offering. Having said that, the 'nagging' here is considerably less intrusive than we have seen in some other free solutions.

Besides our minor quibbles, this is a pretty complete suite solution with only small issues separating it from paid-for products. On top of being a remarkable bargain, it is well designed, pleasantly laid out and intuitive to operate, and has consistently demonstrated excellent stability and reliability, solid detection levels and light performance impact, over several years of testing.

Along with a few close competitors, this seems to be a strong indication of the way the free sector is moving – away from intrusive nagging and crippled, basic products, towards a future of complete, fully functioning, multi-component suite solutions, given away free of charge to a grateful populace. This is a firm challenge to the other vendors to take the next step along the road to complete freedom.

Technical information

Agnitum Outpost Security Suite 7.1 Free Edition was variously tested on:

AMD Phenom, 4GB RAM, running *Microsoft Windows XP Professional SP3 (x32)* and *Windows 7 Professional (x64)*.

Intel Atom 1.6GHz netbook, 2GB RAM, running *Microsoft Windows XP Professional SP3 and Windows 7 Professional*.

COMPARATIVE REVIEW

VBSPAM COMPARATIVE – MAY 2011

Martijn Grooten

When we embarked on VBSpam testing two years ago we had a number of goals, one of which was to provide the anti-spam community – both the developers and the users of spam filters – with information that would help them in the fight against unwanted email.

I still find this one of the most rewarding aspects of my job: developers let me know that they have used feedback from our tests to tweak their filters in order to provide better protection for their customers' inboxes. But with reward comes a responsibility: we have to make sure that the feedback we give is relevant and reflects a real situation.

I have been asked several times why we don't add a stream of newsletters to the test. Although we would like to do this (and, in fact, hope to do so soon), we are very hesitant about incorporating newsletter filtering performance into the criteria for earning a VBSpam award.

There are few email users who do not subscribe to at least a small number of newsletters. But there are even fewer users who have not received unwanted and apparently unsolicited newsletters. These messages may be straightforward spam. However, as a result of legal loopholes and small print in terms and conditions, they may not be spam according to some definitions. In some instances the user may even have subscribed to the newsletters and then forgotten they had done so.

This does not change the user's experience of receiving unwanted email or spam – but there is a very real possibility that exactly the same newsletters are *wanted* by other users. For this reason, we cannot make absolute statements about the right or wrong way to treat newsletters.

As a consequence, the performance numbers we currently report – in particular the spam catch rates – could well be better than those experienced by an organization using the product in a real-world situation. We do not think this should matter. After all, mail traffic can differ greatly between organizations, and so will spam catch rates and false positive rates. Moreover, to fully understand the meaning of a catch rate, one has to know the size of the full inbound mail stream; few people do.

What does matter, though, is the fact that our tests are comparative. This doesn't just mean that we test multiple products under the same circumstances, it also means that the results can and should be compared. A product that blocks 99.70% of spam in our tests might not have the same catch rate when used by a customer, but it is likely to perform better for that customer than a product that only catches 99.10% of spam in our tests.

The 13th VBSpam test did not prove unlucky for any of the participating products, with all 19 entrants receiving a VBSpam award. This report presents the detailed figures that distinguish the good products from the really good ones, but perhaps the most important question for potential customers to ask is: why did other products opt not to be tested?

THE TEST SET-UP

The VBSpam test methodology can be found at http://www.virusbtn.com/vbspam/methodology/. As usual, email was sent to the products in parallel and in real time, and products were given the option to block email pre-DATA. Three products chose to make use of this option.

As in previous tests, the products that needed to be installed on a server were installed on a *Dell PowerEdge R200*, with a 3.0GHz dual core processor and 4GB of RAM. The *Linux* products ran on *SuSE Linux Enterprise Server 11*; the *Windows Server* products ran on either the 2003 or the 2008 version, depending on which was recommended by the vendor.

Two of the virtual products tested ran on *VMware ESXi 4.1*, while two others ran on *VMware Server 2.0*, which we had hitherto used for all virtual products.

To compare the products, we calculate a 'final score', which is defined as the spam catch (SC) rate minus five times the false positive (FP) rate. Products earn VBSpam certification if this value is at least 97:

$$SC - (5 \times FP) \geq 97$$

THE EMAIL CORPUS

The test ran for 16 consecutive days, from 12am GMT on Saturday 9 April 2011 until 12am GMT on Monday 25 April 2011.

The corpus contained 74,746 emails, 72,008 of which were spam. Of these, 40,674 were provided by *Project Honey Pot* and 31,334 were provided by *Abusix*; in both cases, the messages were relayed in real time, as were the 2,738 legitimate emails. As before, the legitimate emails were sent in a number of languages to represent an international mail stream and came from countries all over the world, including India, Russia, El Salvador and Japan.

Figure 1 shows the average catch rate of all full solutions throughout the test. To avoid the average being skewed by poorly performing products, we excluded the highest and lowest catch rate for each hour.

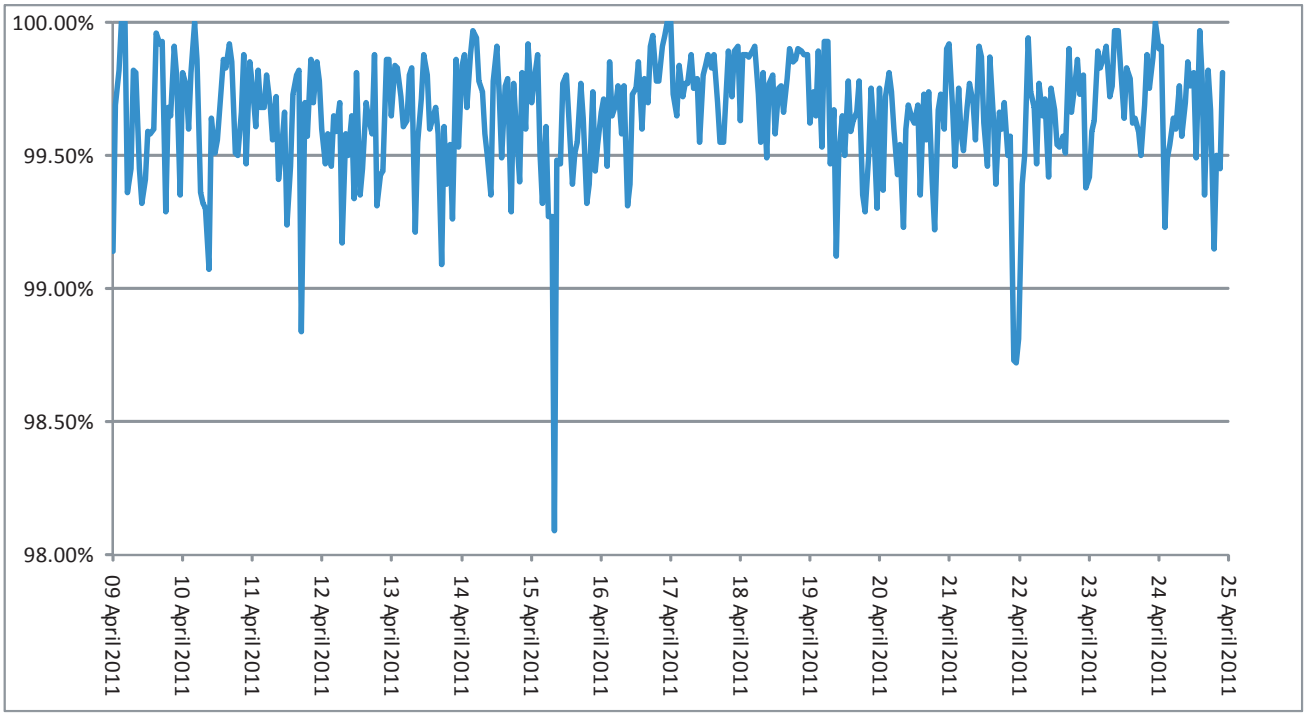


Figure 1: Average spam catch rate of all full solutions throughout the test.

As before, we looked at differences between the full corpus and the subset of 'difficult' spam – defined as those messages missed by at least two different filters; the latter concerned slightly more than 1 in 36 messages, which is a higher ratio than during previous tests.

This time we looked at the uniqueness of the messages in the full corpus of spam. An important characteristic of the vast majority of spam is that it is sent in bulk. Many filters make use of this characteristic to detect spam and it is commonly believed that making spam more 'unique' increases the likelihood that it will stay under the radar and remain uncaught.

We used a rather simple definition of uniqueness, determining two messages to be 'similar' if their subjects were equal[1] and if the number of lines in the raw bodies were equal. Of course, this is by no means the best way to define uniqueness, and an important part of anti-spam research is about finding ways to group similar messages. Still, it is an easy way to test the commonly held belief that unique messages are harder to filter.

In Table 1, on the left it can be seen that 22.2% of all spam was unique according to the above definition. 12.9% of all spam was part of a group of two to five similar messages, and so on; finally, 52.4% of all spam was part of a group of 51 or more messages (the largest group consisted of 615 pill spam messages). On the right-hand side, it is shown that among the 'difficult' spam, 28.5% was unique (within the full corpus), and 38.1% of difficult spam was part of a group of 51 or more messages – indicating that there is some truth in the belief that unique messages are harder to filter.

No. of similar messages	Percentage of spam corpus
1	22.2%
2 to 5	12.9%
6 to 10	3.1%
11 to 20	2.6%
21 to 50	6.8%
51+	52.4%

No. of similar messages	Percentage of 'difficult' spam
1	28.5%
2 to 5	10.5%
6 to 10	4.7%
11 to 20	5.4%
21 to 50	12.8%
51+	38.1%

Table 1: Left: Uniqueness of messages seen in the spam feeds. Right: Uniqueness of spam messages missed by at least two full solutions.

By using more advanced definitions of similarity and a larger corpus, one would be able to obtain more refined

[1] We only used one recipient in the test; hence messages with the recipient's local-part, domain or email address in the subject could still be considered similar.

results and possibly show a stronger correlation between uniqueness and difficulty of filtering.

BLACK- AND WHITELISTING

In our tests, we have always focused on the core of the spam filter: the anti-spam engine.

However, a spam filter comprises much more than its engine. This fact was brought home recently to security company *RSA*, one of whose employees received a targeted spam message containing a malicious attachment. The company's spam filter did its job, putting the message into quarantine; however, this did not stop the user from fishing the email out of quarantine, opening the attachment and thus exposing a backdoor into the company's network.

In this and future tests, we will look at a number of different features of spam filters, reporting on whether the products on test have these features and, if they do, whether they work as intended. We start this month by looking at black- and whitelisting.

We considered four possible features:

- The possibility to whitelist email coming from a certain IP address.
- The possibility to blacklist email coming from a certain IP address.
- The possibility to whitelist email based on the senders' domain[2].
- The possibility to blacklist email based on the senders' domain.

It is very important to note that we do not wish to make assertions as to whether this is something spam filters *should* have. Several participating filters do not provide some or all of these features, and they may have good reasons for not doing so. Using a black- or whitelist incorrectly could lead to many missed emails, or to inboxes overflowing with spam.

Still, for many a system administrator having these options may be just what is needed to allow mail through from an organization whose messages keep being blocked, or to stop newsletters that fail to respond to unsubscribe requests.

We tested these properties by slightly modifying emails that were previously blocked (to test whitelisting) or allowed (to test blacklisting) and resending them. The modifications to the products' settings were made by the testers following the developers' instructions.

[2] We tested by sending emails where both the MAIL FROM address in the SMTP envelope and the From: address in the email headers were on the white- or blacklisted domain.

RESULTS

AnubisNetworks Mail Protection Service

SC rate: 99.92%

FP rate: 0.07%

Final score: 99.55

Project Honey Pot SC rate: 99.88%

Abusix SC rate: 99.96%

AnubisNetworks' R&D recently built a tool[3] that tracks *Twitter* spammers, to demonstrate the company's awareness of the fact that unwanted messages are not restricted to email. This didn't detract from the performance of the spam filter though – the product blocked close to 100 per cent of spam messages and two false positives (the first since September last year) were not enough to get in the way of winning a sixth VBSpam award.

Both black- and whitelisting are possible with this product. The options were easily set in the web interface and worked very well.

BitDefender Security for Mail Servers 3.0.2

SC rate: 99.84%

FP rate: 0.00%

Final score: 99.84

Project Honey Pot SC rate: 99.85%

Abusix SC rate: 99.83%

BitDefender's anti-spam product runs on a number of platforms, but we have been testing the *Linux* product – integrated with *Postfix* – since the very first test. *Linux* fans will be pleased to learn that domain black- and whitelists can be activated by adding the domains to a configuration file. IP black- and whitelisting is not possible, but as the product is an SMTP proxy, blacklisting is usually an option in the ambient SMTP server.

What should please users of the product even more is that it earned yet another VBSpam award and continues to be the only product to have won an award in all 13 VBSpam tests. Moreover, with an excellent spam catch rate and zero false positives, its final score was in the top five of this test.

[3] http://www.tweetspike.org/.

eleven eXpurgate Managed Service 3.2

SC rate: 99.64%
FP rate: 0.00%
Final score: 99.64
Project Honey Pot SC rate: 99.43%
Abusix SC rate: 99.92%

eleven saw a slight improvement in its spam catch rate, but probably more important to the developers is the fact that it did not generate any false positives in this test – and has not done so in its last three tests. The company can thus pride itself on a third VBSpam award.

The web interface that comes with the hosted solution allows for black- and whitelisting of both IPs and domains, and setting them was a trivial task.

Fortinet FortiMail

SC rate: 99.82%
FP rate: 0.00%
Final score: 99.82
Project Honey Pot SC rate: 99.76%
Abusix SC rate: 99.91%

A regular VBSpam participant and repeated award winner, this month *Fortinet* wins its 12th consecutive VBSpam award. *FortiMail* equalled last month's spam catch rate and once again achieved a zero false positive rate, giving it the same high final score.

The web interface that is used to control the appliance allows for both black- and whitelisting; setting it up and getting it to work presented few problems.

GFI MailEssentials

SC rate: 99.62%
FP rate: 0.51%
Final score: 97.07
Project Honey Pot SC rate: 99.52%
Abusix SC rate: 99.76%

GFI is not new to the VBSpam tests, having won several VBSpam awards with *VIPRE*, a product originally developed by *Sunbelt*. However, *MailEssentials* is the Maltese company's own in-house-developed product. It runs on *Windows* and hooks into a number of SMTP servers, including *Exchange* and *IIS*. We tested it using the latter.

Set-up was easy and the user interface is clear and intuitive. It can be used to manage domain black- and whitelists and IP whitelists, all of which worked in a pretty straightforward manner. The product does not allow for IP blacklisting, but the developers recommend customers use the ambient SMTP server for that.

The product did well at catching spam, but generated twice as many false positives as the next most poorly performing product. *MailEssentials* did just scrape a high enough final score to win a VBSpam award, but it will now be up to *GFI*'s developers to show that the high false positive rate can be lessened by making some modifications to the product and/or its settings.

Halon Mail Security

SC rate: 99.46%
FP rate: 0.00%
Final score: 99.46
Project Honey Pot SC rate: 99.78%
Abusix SC rate: 99.06%

I have said it might take a test or two for products to fully adapt to our set-up and environment, but *Halon Mail Security* proved me wrong in the last test. The Swedish product (we tested the virtual appliance) combined a rather good spam catch rate with zero false positives on its first entry, and repeated the achievement in this test, winning its second VBSpam award.

Unsurprisingly, given the scripting language that can be used to tweak the product, the addition of black- and whitelists is possible – and these checks can take place at different places during the transaction. We chose the most straightforward places (less tech-savvy users will be pleased to know that no scripting was involved) and found them to work well.

Kaspersky Anti-Spam 3.0

SC rate: 99.37%
FP rate: 0.00%
Final score: 99.37
Project Honey Pot SC rate: 99.46%
Abusix SC rate: 99.25%

	True negatives	False positives	FP rate	False negatives	True positives	SC rate	Final score
AnubisNetworks	2736	2	0.07%	58	71950	99.92%	99.55
BitDefender	2738	0	0.00%	117	71891	99.84%	99.84
eleven	2738	0	0.00%	256	71752	99.64%	99.64
FortiMail	2738	0	0.00%	127	71881	99.82%	99.82
GFI MailEssentials	2724	14	0.51%	271	71737	99.62%	97.07
Halon Mail Security	2738	0	0.00%	386	71622	99.46%	99.46
Kaspersky Anti-Spam	2738	0	0.00%	454	71554	99.37%	99.37
Libra Esva	2738	0	0.00%	40	71968	99.94%	99.94
McAfee Email Gateway	2738	0	0.00%	57	71951	99.92%	99.92
McAfee EWS	2737	1	0.04%	1054	70954	98.54%	98.35
OnlyMyEmail	2738	0	0.00%	2	72006	100.00%	100.00
Sophos Email Appliance	2737	1	0.04%	169	71839	99.77%	99.58
SPAMfighter	2734	4	0.15%	213	71795	99.70%	98.97
SpamTitan	2731	7	0.26%	46	71962	99.94%	98.66
Symantec Messaging Gateway	2738	0	0.00%	74	71934	99.90%	99.90
The Email Laundry	2736	2	0.07%	168	71840	99.77%	99.40
Vade Retro	2736	2	0.07%	1336	70672	98.14%	97.78
Vamsoft ORF	2738	0	0.00%	417	71591	99.42%	99.42
Spamhaus*	2738	0	0.00%	799	71209	98.89%	98.89

*As the only partial solution in this test, the results for *Spamhaus* are listed separately from the full solutions.
(Please refer to text for full product names.)

Since installing *Kaspersky Anti-Spam* two years ago, I have not needed to look at the web interface and I had almost even forgotten what kind of interface the product used. That, of course, is a good thing as it demonstrates that the product has been running without issues. On revisiting the interface, to add black- and whitelists, I found it to be intuitive and easily navigated.

It was good to see that last month's drop in performance was a one-off affair, with the spam catch rate returning to well over 99%, still with no false positives. This performance easily wins *Kaspersky* its 11th VBSpam award.

Libra Esva 2.0

SC rate: 99.94%
FP rate: 0.00%
Final score: 99.94
Project Honey Pot SC rate: 99.95%
Abusix SC rate: 99.94%
SC rate pre-DATA: 98.64%

Prior to this test, we moved *Libra Esva*'s virtual product to our new *VMware ESXi 4.1* server. The move was easy, thanks to good work from both the product's and *VMware*'s developers. The product's simple web interface allows for the

use of IP and domain black- and whitelists, but domain whitelisting takes place after the SMTP traffic is checked against the IP blacklists used by the product, and thus might not work in all cases.

Users may have little need for domain whitelisting though. With zero false positives and a 99.94% spam catch rate, *Libra Esva* wins its seventh VBSpam with the second highest final score for the third time in a row.

McAfee Email Gateway (formerly IronMail)

SC rate: 99.92%
FP rate: 0.00%
Final score: 99.92
Project Honey Pot SC rate: 99.87%
Abusix SC rate: 99.98%

Like most products, *McAfee*'s *Email Gateway* appliance is controlled by a web interface. I had not looked at the interface for some time, and was impressed by its many bells and whistles. Black- and whitelisting are possible, though less straightforward than with most products. However, given the huge consequences mistakes can have, making sure users really know what they're doing might not be a bad thing.

Of course, if the product performs its main task well there should be little need for black- and whitelisting. This is certainly the case for the *Email Gateway* appliance, and with a spam catch rate of 99.92% and zero false positives (down from six), it wins its 11th consecutive VBSpam award with the third highest final score.

McAfee Email and Web Security Appliance

SC rate: 98.54%
FP rate: 0.04%
Final score: 98.35
Project Honey Pot SC rate: 98.73%
Abusix SC rate: 98.28%

I've always been charmed by the web interface of *McAfee*'s *EWS* appliance with its many options and, unsurprisingly, it let me add black- and whitelists easily.

However, the product's developers will be more concerned with its performance, given that it failed to win a VBSpam award in the last test. Happily, *EWS*'s performance improved significantly – this time generating only a single false positive. There is still some room for improvement, but the product nevertheless wins its tenth VBSpam award.

OnlyMyEmail's Corporate MX-Defender

SC rate: 100.00%
FP rate: 0.00%
Final score: 100.00
Project Honey Pot SC rate: 100.00%
Abusix SC rate: 99.99%

To use black- and whitelists, users of *OnlyMyEmail*'s *MX-Defender* need to fill out a form on the company's website. I received a quick response to this and soon realised the benefit of a second check of my request: I had made a mistake in filling out the form and was asked if I really wanted what I had asked for. After clarification, the black- and whitelists were added. The IP whitelist does not always work though, and blocked a very spammy test message from the whitelisted IP – which hints towards the fact that pure whitelisting is not always a good idea.

But with zero false positives, *OnlyMyEmail*'s users will find little need to use whitelisting. And they will not need to add many items (if any) to the blacklist either, as the product missed just two out of over 70,000 spam messages. In fact, not only did the product achieve the highest spam catch rate for the fourth time in a row, it also achieved the highest final score in the test for the second time and, rounded to 100, the highest final score and spam catch rate since our tests began two years ago. Needless to say, *OnlyMyEmail* can be extremely proud of its fourth VBSpam award.

Sophos Email Appliance

SC rate: 99.77%
FP rate: 0.04%
Final score: 99.58
Project Honey Pot SC rate: 99.77%
Abusix SC rate: 99.75%

In March, *Sophos* won its eighth VBSpam award in as many tests and this month it easily adds a ninth to its tally. A slight drop in performance is not a serious issue for the appliance – which achieved the highest final score in the previous test.

The simple web interface allows for both black- and whitelisting by domain and IP address and all worked as expected.

	Project Honeypot		Abusix		pre-DATA†						
	False negative	SC rate	False negative	SC rate	False negative	SC rate	STDev‡	IP WL	IP BL	Dom WL	Dom BL
AnubisNetworks	47	99.88%	11	99.96%			0.21	+	+	+	+
BitDefender	63	99.85%	54	99.83%			0.39	-	-	+	+
eleven	230	99.43%	26	99.92%			0.94	+	+	+	+
FortiMail	99	99.76%	28	99.91%			0.36	+	+	+	+
GFI MailEssentials	197	99.52%	74	99.76%			0.49	+	-	+	+
Halon Mail Security	91	99.78%	295	99.06%			0.76	+	+	+	+
Kaspersky Anti-Spam	218	99.46%	236	99.25%			1.12	+	+	+	+
Libra Esva	22	99.95%	18	99.94%	28949	98.64%	0.17	+	+	+	+
McAfee Email Gateway	51	99.87%	6	99.98%			0.21	+	+	+	+
McAfee EWS	515	98.73%	539	98.28%			1.48	+	+	+	+
OnlyMyEmail	0	100.00%	2	99.99%			0.04	-	+	+	+
Sophos Email Appliance	92	99.77%	77	99.75%			0.51	+	+	+	+
SPAMfighter	106	99.74%	107	99.66%			0.57	+	+	+	+
SpamTitan	20	99.95%	26	99.92%			0.19	-	+	+	+
Symantec Messaging Gateway	56	99.86%	18	99.94%			0.23	+	+	+	+
The Email Laundry	135	99.67%	33	99.89%	29249	99.30%	0.35	-	-	+	+
Vade Retro	352	99.13%	984	96.86%			2.51	-	-	-	-
Vamsoft ORF	320	99.21%	97	99.69%			0.60	+	+	+	+
Spamhaus*	557	98.63%		99.23%	30740	98.15%	1.05	-	-	-	-

† pre-DATA filtering was optional and was applied on the full spam corpus. All of *The Email Laundry*'s false positives occurred pre-DATA; none of the other products had pre-DATA false positives.

‡ The standard deviation of a product is calculated using the set of its hourly spam catch rates.

* As the only partial solution in this test, the results for *Spamhaus* are listed separately from the full solutions.

(Please refer to text for full product names.)

SPAMfighter Mail Gateway

SC rate: 99.70%

FP rate: 0.15%

Final score: 98.97

Project Honey Pot SC rate: 99.74%

Abusix SC rate: 99.66%

The simple web interface provided for system administrators to modify *SPAMfighter*'s settings made it a simple process to find and add black- and whitelists.

Not only do the developers deserve praise for that, but even more so for the fact that the product achieved its highest spam catch rate to date. With just four false positives, this also gave the product its highest final score and *SPAMfighter* earns its tenth VBSpam award.

SpamTitan

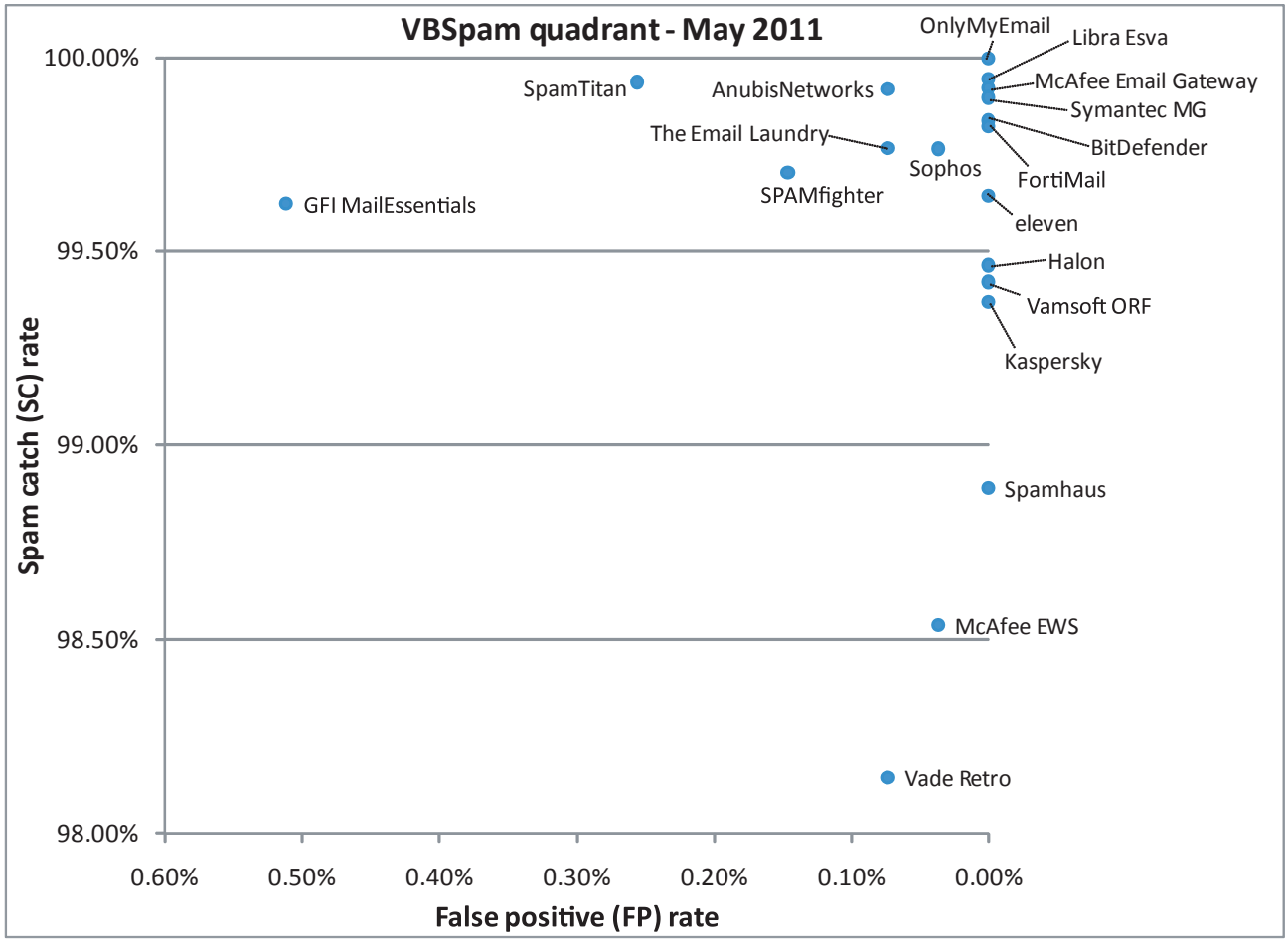
SC rate: 99.94%

FP rate: 0.26%

Final score: 98.66

Project Honey Pot SC rate: 99.95%

Abusix SC rate: 99.92%



(Please refer to text for full product names.)

This month's test marks *SpamTitan*'s tenth entry and also the product's tenth VBSpam award. It had been some time since I last looked at the interface, but on using it to add black- and whitelists I remembered how easily it worked. Whitelisting by IP, however, is not available.

As in previous tests, *SpamTitan* blocked close to 100 per cent of all spam, though the second highest false positive rate in this month's test means there is something that can be improved upon.

Symantec Messaging Gateway 9.5 powered by Brightmail

SC rate: 99.90%

FP rate: 0.00%

Final score: 99.90

Project Honey Pot SC rate: 99.86%

Abusix SC rate: 99.94%

Symantec Messaging Gateway 9.5 powered by Brightmail (formerly *Symantec Brightmail Gateway*) is the newest version of the product, and we used the change in product version as an opportunity to move it to a *VMware ESXi 4.1* virtual server. The 9.5 version includes what the developers believe to be improvements to the anti-spam engine and I was curious to see if this would be reflected in the product's performance. It was: the spam catch rate – which was already excellent – was improved upon, and there were no false positives this time. The product thus wins its ninth VBSpam award in as many tests with the fourth highest final score.

What has not changed – at least not in a noticeable way – is the web interface to control the (virtual) appliance, and I was rather pleased by that. Adding IP and domain black- and whitelists and telling the product to do specific things when these lists were triggered was easy and worked as expected.

Products ranked by final score	Final score
OnlyMyEmail	100.00
Libra Esva	99.94
McAfee Email Gateway	99.92
Symantec Brightmail Gateway	99.90
BitDefender	99.84
FortiMail	99.82
eleven	99.64
Sophos Email Appliance	99.58
AnubisNetworks	99.55
Halon Security	99.46
ORF	99.42
The Email Laundry	99.40
Kaspersky Anti-Spam	99.37
SPAMfighter	98.97
Spamhaus	98.89
SpamTitan	98.66
McAfee EWS	98.35
Vade Retro	97.78
GFI MailEssentials	97.07

The Email Laundry

SC rate: 99.77%
FP rate: 0.07%
Final score: 99.40
Project Honey Pot SC rate: 99.67%
Abusix SC rate: 99.89%
SC rate pre-DATA: 99.30%

The Email Laundry does not allow customers to black- or whitelist by IP address and it should be noted once again that we do not wish to make assertions about whether or not this is a good thing; it is certainly something where human mistakes can have huge consequences. Domain black- and whitelisting is possible though, and was easily added in the product's interface.

As before, the product caught a large amount of spam, the vast majority of which was blocked at the SMTP level. As both the spam catch rate and the false positive rate improved, so did the final score and the product easily won its seventh VBSpam award.

Vade Retro Center

SC rate: 98.14%
FP rate: 0.07%
Final score: 97.78
Project Honey Pot SC rate: 99.13%
Abusix SC rate: 96.86%

Vade Retro offers a wide range of solutions, from hardware and virtual appliances to a number of hosted solutions. Unlike most of the other solutions, the hosted solution we have been testing does not allow for IP or domain black- and whitelisting.

The product scored well enough to achieve its seventh VBSpam award, but its developers may want to look at improving its spam catch rate – which was lower than any other product in this test. The fact that this was largely due to problems with the *Abusix* corpus may help them find the reason for this drop in performance.

Vamsoft ORF

SC rate: 99.42%
FP rate: 0.00%
Final score: 99.42
Project Honey Pot SC rate: 99.21%
Abusix SC rate: 99.69%

I have sung the praises of *ORF*'s user interface before, and using it to add black- and whitelists was once again a pleasure. Given the fact that *ORF* had no false positives for the fifth time in seven tests, though, few people are likely to need the whitelisting options.

While keeping the false positives to zero, *ORF* also managed to improve its spam catch rate, which was higher than in any previous test. A seventh VBSpam award will be proudly received at the company's Hungarian headquarters.

Spamhaus ZEN+DBL

SC rate: 98.89%
FP rate: 0.00%

Spamhaus ZEN+DBL contd.

Final score: 98.89
Project Honey Pot SC rate: 98.63%
Abusix SC rate: 99.23%
SC rate per-DATA: 98.15%

The increasing occurrence of URL shorteners in spam messages has presented *Spamhaus*'s *DBL* blocklist with a problem: blocking them would give false positives on the legitimate use of such shorteners; allowing them would give spammers a way to include their malicious URLs while avoiding detection.

The blocklist's developers have come up with a rather neat solution, returning different codes for known shorteners. This prevents messages containing them from being marked as spam, but allows users of the blacklists to resolve the real URL and hold this against the blacklist. (We did not do this in our tests.)

This ability to constantly adapt to spammers' techniques has earned *Spamhaus* eight VBSpam awards already and it adds a ninth to its tally in this test, catching a higher percentage of spam than on any previous occasion.

(As *Spamhaus* is only a partial solution, which needs to be integrated into a full solution, it does not make sense to black- or whitelist within this product.)

CONCLUSION

After two years of testing, and VBSpam certifications being awarded to 27 different products, readers should by now have a good picture of which products provide decent inbox protection and, more importantly, which of those can provide protection reliably over a prolonged period. We will, of course, continue to test products and award VBSpam certifications, but we also intend to provide more information about the products we look at.

The black- and whitelisting tests we introduced this month took some time to set up, but I think it was worth doing: with so many products performing so well, customers might want to differentiate between products by looking into the availability (or otherwise) of certain extras. We intend to look into more of these additional features in future tests.

The next VBSpam test will run in June 2011, with the results scheduled for publication in July. Developers interested in submitting products should email martijn.grooten@virusbtn.com.

END NOTES & NEWS

The 20th Annual EICAR Conference will be held 9–10 May 2011 in Krems, Austria. This year's conference is named 'New trends in malware and anti-malware techniques: myths, reality and context'. For full details see http://www.eicar.org/conference/.

The 6th International Conference on IT Security Incident Management & IT Forensics will be held 10–12 May 2011 in Stuttgart, Germany. See http://www.imf-conference.org/.

TakeDownCon takes place 14–19 May 2011 in Dallas, TX, USA. The event aims to bring together security researchers from corporate, government and academic sectors as well the underground to present and debate the latest security threats and disclose and scrutinize vulnerabilities. For more details see http://www.takedowncon.com/.

The 2nd VB 'Securing Your Organization in the Age of Cybercrime' Seminar takes place 24 May 2011 in Milton Keynes, UK. Held in association with the MCT Faculty of The Open University, the seminar gives IT professionals an opportunity to learn from and interact with security experts at the top of their field and take away invaluable advice and information on the latest threats, strategies and solutions for protecting their organizations. For details see http://www.virusbtn.com/seminar/.

CONFidence 2011 takes place 24–25 May 2011 in Krakow, Poland. Details can be found at http://confidence.org.pl.

The 2011 National Information Security Conference will be held 8–10 June 2011 in St Andrews, Scotland. Registration for the event is by qualification only – applications can be made at http://www.nisc.org.uk/.

The 23rd Annual FIRST Conference takes place 12–17 June 2011 in Vienna, Austria. The conference promotes worldwide coordination and cooperation among Computer Security Incident Response Teams. For more details see http://conference.first.org/.

SOURCE Seattle 2011 will be held 16–17 June 2011 in Seattle, WA, USA. For more details see http://www.sourceconference.com/.

The Eighth Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2011) takes place 7–8 July 2011 in Amsterdam, The Netherlands. For details see http://www.dimva.org/dimva2011/.

Black Hat USA takes place 30 July to 4 August 2011 in Las Vegas, NV, USA. DEFCON 19 follows the Black Hat event, taking place 4–7 August, also in Las Vegas. For more information see http://www.blackhat.com/ and http://www.defcon.org/.

The 20th USENIX Security Symposium will be held 10–12 August 2011 in San Francisco, CA, USA. See http://usenix.org/.

The 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS 2011) will be held in Perth, Australia 1–2 September, 2011. See http://ceas2011.debii.edu.au/.

VB2011 takes place 5–7 October 2011 in Barcelona, Spain. For full programme details including abstracts for each paper, and online registration see http://www.virusbtn.com/conference/vb2011/.

RSA Europe 2011 will be held 11–13 October 2011 in London, UK. For details see http://www.rsaconference.com/2011/europe/index.htm.

Hacker Halted 2011 will take place 21–27 November in Miami, FL, USA. See http://www.hackerhalted.com/2011/.

The sixth annual APWG eCrime Researchers Summit will be held 7–9 November 2011 in San Diego, CA, USA. The summit will bring together academic researchers, security practitioners and law enforcement to discuss all aspects of electronic crime and ways to combat it. For more details see http://www.antiphishing.org/ecrimeresearch/2011/cfp.html.

The CSI 2011 Annual Conference will be held 6–11 November 2011 in Washington D.C., USA. See http://www.CSIannual.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)1865 543153
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/