

virus

BULLETIN

Fighting malware and spam

CONTENTS

- 2 **COMMENT**
The good, the bad and the blurring boundaries
- 3 **NEWS**
AusCERT delegates get more than they bargained for
Could do better all round
- 3 **VIRUS PREVALENCE TABLE**
- FEATURES**
- 4 Metafile art class
- 8 Strike me down, and I shall become more powerful!
- 11 System cleaning: getting rid of malware from infected PCs
- 15 **CONFERENCE REPORT**
EICAR 2008, c'était merveilleux!
- 16 **COMPARATIVE REVIEW**
Ubuntu Linux 8.04LTS Server Edition
- 26 **END NOTES & NEWS**

IN THIS ISSUE

ARTS AND CRAFTS

Like its predecessor the Windows Metafile Format, the Enhanced Metafile Format has proven to be susceptible to misappropriation. Dennis Elser provides an in-depth description of a recent remotely exploitable file format vulnerability within the Windows graphics device interface (GDI).
page 4

THE ROOTKIT STRIKES BACK

The rootkit that has been dubbed one of the stealthiest ever seen in the wild is back – with improved defences and new stealth code. Aditya Kapoor and Rachit Mathur look at recent developments in the MBR rootkit.
page 8

VB100 ON UBUNTU LINUX

John Hawes dusts off his Linux skills for a comparative review of anti-malware products on the Ubuntu Server platform.
page 16



vb Spam supplement

This month: anti-spam news and events, and Jonathan Zdziarski questions whether your spam filter is really adaptive.



'The reality is that the distinction between legitimate and malicious software is an ever blurring line.'

Greg Day, McAfee

THE GOOD, THE BAD AND THE BLURRING BOUNDARIES

In April this year I attended the Council of Europe's (CoE) cooperation against cybercrime conference in Strasbourg (<http://www.coe.int/cybercrime>). The goals of the event were to review the effectiveness of existing legislation on cybercrime (which is currently signed by 44 countries and ratified by 22 of them) and prepare proposals for improvements to it. A total of 65 countries were represented at the event, the majority of attendees being from either legal, law enforcement or government backgrounds. The CoE does outstanding work in attempting to standardize the laws relating to cybercrime, and trying to reduce the number of countries in which cybercriminals can hide.

One aspect that remains a very significant challenge is that of capture of evidence. During the conference specific focus was given to encouraging Internet Service Providers (ISPs) to work more closely with law enforcement agencies to provide the necessary support. But, within the UK, ISPs have also been feeling pressure from government to monitor and control copyrighted content being downloaded through means such as P2P sharing. Under the Regulation of Investigatory Powers Act (RIPA), ISPs can only inspect data packets when acting under authority, so it would seem that in the UK the greater involvement of ISPs in monitoring Internet use is untenable without additional modifications to the law (<http://www.guardian.co.uk/technology/2008/feb/22/filesharing>).

Editor: Helen Martin

Technical Consultant: John Hawes

Technical Editor: Morton Swimmer

Consulting Editors:

Nick FitzGerald, *Independent consultant, NZ*

Ian Whalley, *IBM Research, USA*

Richard Ford, *Florida Institute of Technology, USA*

Complicating the matter still further for ISPs are the 'value-add' services such as smart advertising (e.g. *NebuAd*, *Adzilla* and *FrontPorch*) that businesses are looking to offer to increase their revenue potential. The result is a dichotomy of pressures and requirements to monitor whilst also tracking user behaviour and carefully trying not to infringe on privacy.

Conversely, the criminal elements are attempting to legitimize their software, often hiding behind EULAs and selling their tools under the auspices of 'for educational purposes only', thus avoiding the law enforcement radar. The reality is that the distinction between legitimate and malicious software is an ever blurring line, with research teams needing legal expertise as they try to define all the greyware in between.

With all of these factors and new commercial tools we are heading for a collision in the greyware space. Over the last few months there has been much discussion about the boundaries of commercial software, especially in terms of user privacy. *McAfee* defines spyware as 'software whose function includes transmitting personal information to a third party without the user's knowledge or consent,' continuing: 'this usage is distinct from the common usage of spyware to represent commercial software that has security or privacy implications.'

In a recent trial of an online advertising system in the UK the media highlighted that users were not notified that a cookie was being installed on their systems (<http://news.bbc.co.uk/1/hi/technology/7325451.stm>). Some have argued that this pushes the system in question into the category of spyware. The case has been a wakeup call for many, highlighting that the challenge is in how the software is presented to the end user – in other words it is an issue of user awareness and consent. Just as we are given the option to opt in to receiving marketing emails, users should be aware of the software installed on their systems to give them smart advertising.

With further trials planned soon, and increasing numbers of similar tools becoming available, cooperation between ISPs/implementers, vendors and the security industry must ensure that such tools are implemented in a way that guarantees they are classified correctly. Yet, as the boundaries continue to blur, this will remain a hotly debated subject.

As the volume of greyware/potentially unwanted programs continues to grow, I have to wonder how long it will be before we have more lawyers than malware researchers. Indeed, today it can take longer to comprehend the legal stance on a piece of code than it does to perform the analysis. The bad guys will continue to sail close to the wind, and the good guys must be careful!

NEWS

AUSCERT DELEGATES GET MORE THAN THEY BARGAINED FOR

A few red faces were to be seen at last month's AusCERT security conference held on the Gold Coast in Australia, and not through over exposure to the sun. Infected USB sticks were inadvertently handed out to attendees at one of the event's scheduled tutorials. AusCERT's marketing manager Claire Groves told *Search Security* that it was Australian telecoms company *Telstra* that made the faux pas, but that 'as soon as they found out [that the sticks were infected] they recalled them'. One imagines the *Telstra* representatives are still finding remnants of egg on their faces.

COULD DO BETTER ALL ROUND

The European Union's information security body has warned that many countries in Europe need to pull their socks up when it comes to information security, and called for changes in legislation that will make the reporting of security breaches by businesses mandatory.

The European Network and Information Security Agency (ENISA) highlighted in its General Report 2007, published last month, that while 14 EU member states have government-supported response teams, many member countries are not equipped to deal with cyber attacks – and leave themselves vulnerable to what it called a 'digital 9/11'.

ENISA also called for the introduction of laws that would force businesses to reveal when the security of their computer systems has been breached. There is currently no requirement for companies to reveal that a breach has taken place – and many businesses avoid reporting such incidents in an attempt to protect their reputations – but the withholding of information about security breaches both leads the public into a false sense of security and makes the task of fighting cybercrime significantly harder.

Andrea Pirotti, executive director of ENISA, said in a statement: 'Europe must take security threats more seriously and invest more resources in NIS [network and information security].'

Meanwhile, in the US an annual report card revealed that federal agencies showed better adherence to information security rules in 2007 than in the previous year, but that nine of the 24 agencies still failed to comply with the rules to a satisfactory degree. The report card assigns a grade to each government agency for its compliance with the Federal Information Security Management Act of 2002. Overall, a 'C' grade was awarded for the combined governmental effort – which was a small step up from last year's 'C-'. However, nine of the agencies were graded 'D' and below – a definite case of 'could do better'.

Prevalence Table – April 2008

Malware	Type	%
NetSky	Worm	21.17%
Cutwail/Pandex/Pushdo	Trojan	16.47%
OnlineGames	Trojan	15.72%
Mytob	Worm	12.37%
Virut	Virus	8.15%
Mydoom	Worm	5.30%
Bagle	Worm	3.80%
Small	Trojan	3.34%
Zafi	Worm	3.09%
Agent	Trojan	2.85%
Grew	Worm	1.67%
Zlob/Tibs	Trojan	1.15%
Stration/Warezov	Worm	0.64%
Sality	Virus	0.64%
Bugbear	Worm	0.56%
Autorun	Worm	0.41%
Klez	Worm	0.31%
VB	Worm	0.27%
Brontok/Rontokbro	Worm	0.23%
Grum	Worm	0.23%
Bagz	Worm	0.17%
Nahata	Worm	0.16%
Bifrose/Pakes	Trojan	0.12%
Delf	Trojan	0.12%
Parite	Worm	0.08%
Feebs	Worm	0.07%
Alman	Worm	0.07%
Bolzano	Virus	0.06%
Lineage/Magania	Trojan	0.06%
Hybris	Worm	0.05%
Nimda	Worm	0.05%
Plexus	Worm	0.05%
Womble	Worm	0.05%
Others ^[1]		0.52%
Total		100.00%

^[1]Readers are reminded that a complete listing is posted at <http://www.virusbtn.com/Prevalence/>.

FEATURE 1

METAFILE ART CLASS

Dennis Elser

Secure Computing Corporation, Germany

Just like its predecessor the Windows Metafile Format (WMF), the Enhanced Metafile Format (EMF) consists of descriptive commands for drawing an image rather than bitplanes of the rendered image itself. And just like malformed WMF files [1], Enhanced Metafiles have also proved to be susceptible to misappropriation.

This article provides a technical analysis of a recent remotely exploitable file format vulnerability within Windows' graphics device interface (GDI) [2].

PAINT BY NUMBERS

The idea behind WMF and EMF files is application and device independence, respectively: metafiles contain a sequence of records that guide a drawing device in how to render an image.

There is no serious difference in functionality between the two file formats other than EMF being truly device independent by maintaining its dimensions, shape and proportions [3].

All EMF records begin with a 32-bit field which is used to identify the record type and another 32-bit field for the size of the record; both fields are in Little-Endian byte order. Next, there is record-specific data of variable length. An EMF image always starts with a header record (type = 0x1),

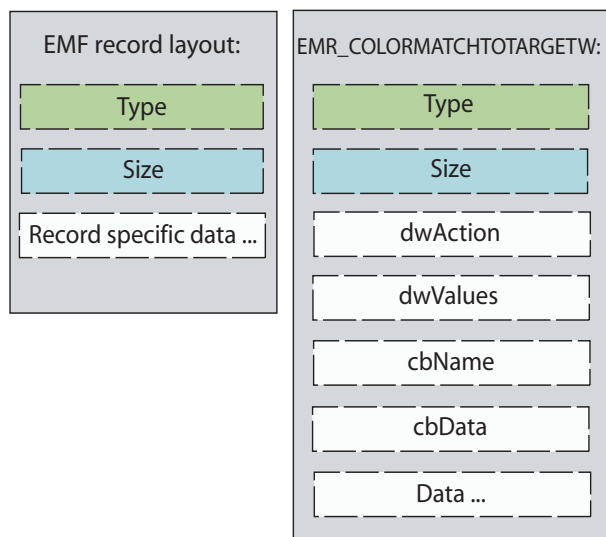


Figure 1: General structure of EMF records and the EMR_COLORMATCHTOTARGETW record.

followed by a sequence of EMF data records and an end-of-file record (type = 0xE).

A record's 'Size' field, as shown in Figure 1, is used by the parser as an offset to find the beginning of the next record (current record's file offset + 'Size'). However, there have also been EMF records with 'overlapping' data, as was the case with the first EMF exploit (MD5: 7DB16FD50CF76 CEF3d29DE47239C1F9A), which was found in the wild only two days after Microsoft published security bulletin MS08-021. Overlapping data wasn't relevant for the exploit's success, but was probably carelessness on the part of the exploit's author and a sign of intentionally corrupted data – easy to spot.

IMAGE INTERPRETATION

We can see three records in the exploit's hex-dump shown in Figure 2 below. Its green markers show the record's type; blue ones show the record's length. The exploit's first record type is EMR_HEADER (0x1), one of approximately 122 different record types. Its second is EMR_COLORMATCHTOTARGETW (0x79).

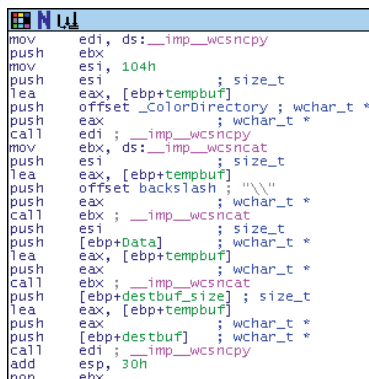
The exploit's third record (0xF1CF7512) is invalid, since the highest record-type number defined on current NT-based machines is 0x7A (wingdi.h). As we will see later, the reason for this is the second record's data exceeding its stated 'Size'.

```

0000 01000000 9C000000 8DFFFFFF B6FFFFFF
0010 E0E00000 02170000 00000000 00000000
0020 FC510000 09740000 20454D46 00000100
0030 A4020000 2F010000 03000000 17000000
0040 6C000000 00000000 96120000 A11A0000
0050 C9000000 21010000 00000000 00000000
0060 00000000 DC120300 3D670400 00005600
0070 43006C00 61007300 73000000 50007200
0080 69006E00 74002000 70007200 65007600
0090 69006500 77000000 00000000 79000000
00A0 50000000 01000000 38000000 00000000
00B0 38000000 EB115B33 C966B985 014B8034
00C0 0B99E2FA 43FFE3E8 EAF00000 F9FD38A9
00D0 99999912 D9955E9F C1C1C1C4 5F9F5A12
00E0 E9853412 E9911875 999D9999 1275CFF1
00F0 17D79775 71639999 9910DC9D CFF1BC29
0100 665B7175 99999910 DC95CFF1 ...
    
```

Figure 2: Hex-dump of an EMF exploit.

By interpreting the structure of record type EMR_COLORMATCHTOTARGETW, which begins at file offset 0x9C in the exploit's hex-dump above, we can find out what causes the overflow. Each of the structure's fields has a size of 32 bits, with the 'Data' field being the only exception: it is an array of bytes that holds a Unicode name of a colour profile with additional raw colour profile data



```

mov     edi, ds:___imp_wcsncpy
push   ebx
mov     esi, 104h
push   esi
lea     eax, [ebp+tempbuf]
push   eax
push   offset _ColorDirectory; wchar_t *
call   edi; ___imp_wcsncpy
mov     ebx, ds:___imp_wcsncat
push   esi
push   esi; size_t
lea     eax, [ebp+tempbuf]
push   eax
push   offset backslash; "\\\"
call   ebx; ___imp_wcsncat
push   esi; size_t
push   [ebp+Data]; wchar_t *
lea     eax, [ebp+tempbuf]
push   eax; wchar_t *
call   ebx; ___imp_wcsncat
push   [ebp+destbuf_size]; size_t
lea     eax, [ebp+tempbuf]
push   eax; wchar_t *
push   [ebp+destbuf]; wchar_t *
call   edi; ___imp_wcsncpy
add     esp, 30h
pop     ebx

```

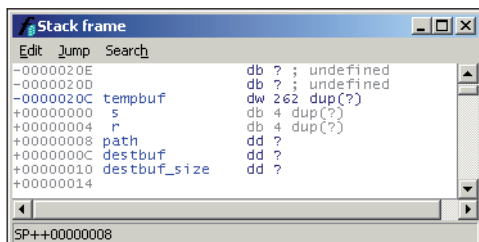
Figure 3: Broken implementation of 'BuildIcmProfilePath'.

appended. A colour profile is a file that contains information about the conversion of colours in the context of a specific device [3].

In *Microsoft's* specification, the size of a 'Data' field in bytes is appointed by the sum of the 'cbName' (0x0) and 'cbData' (0x38) fields, which in the exploit's case results in 0x38. So apparently, the exploit's 'Data' array at file offset 0xB4 is reserved; 0 bytes for a colour profile name but 0x38 bytes for raw colour profile data. However, internal handling of this array of bytes looks a little different, as we will see below.

As soon as the *Windows* GDI parses an EMF file, its size fields are sanity checked (the sum of 'cbName' and 'cbData' must not int-overflow). Depending on the result of these checks, the particular record is flagged as either good or bad internally and only good ones are allowed further processing.

Once an EMR_COLORMATCHTOTARGETW record has passed this test, its 'Data' buffer is processed by 'BuildIcmProfilePath()', which is the function responsible for building a temporary colour profile's path on stack. However, the whole 'Data' buffer is interpreted as a Unicode string by the function without considering the size limitations provided by the 'Size' and 'cbName' fields. So even given a record length of 0x50, the



Address	Disassembly	Comment
-0000020E	db ?	undefined
-0000020D	db ?	undefined
-0000020C	tempbuf	dw 262 dup(?)
+00000000	s	db 4 dup(?)
+00000004	r	db 4 dup(?)
+00000008	path	dd ?
+0000000C	destbuf	dd ?
+00000010	destbuf_size	dd ?
+00000014		

Figure 4: Stackframe of 'BuildIcmProfilePath'.

GDI's EMF parser keeps reading beyond the EMR_COLORMATCHTOTARGETW record's limits until a null-termination character is found within the 'Data' Unicode string.

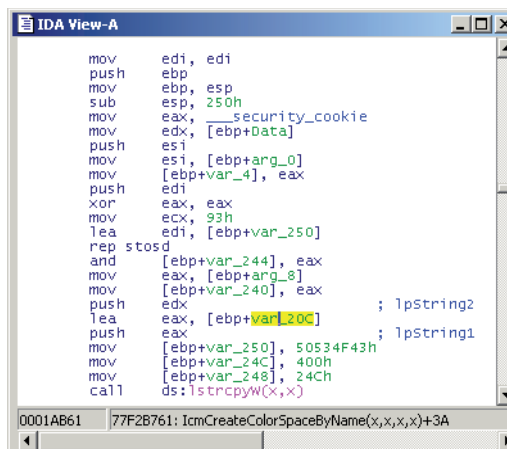
ART FORGERY DETECTION

As both its partial disassembly (Figure 3) and stackframe (Figure 4) show, the maximum length of a path constructed by 'BuildIcmProfilePath()' was supposed to be restricted to 0x104 wide characters (520 bytes). However, due to inappropriate use of `wcsncat()`, the actual maximum restriction is 3 * 0x104 wide characters in theory (1560 bytes), which is far more than can be stuffed into 'tempbuf'. 'tempbuf' can hold a maximum of 262 wide characters including the null-terminating character.

This particular bug is found in early versions of `gdi32.dll` (i.e. *XP SP0*). Later versions of `gdi32.dll` (pre-MS08-021) succeed in calculating the remaining space of 'tempbuf' using `wcslen()`, but still fail to prevent the stack from being overwritten with 'Data' in a subsequent `lstrcpyW()` call found in 'IcmCreateColorSpaceByName()' – which reflects the vulnerability that has been fixed with MS08-021. Starting with that latest patch, the maximum number of wide characters copied to the stack is limited to 0x104.

In the end, with this limitation in effect, *Microsoft* deprecated its own EMF specification, since 'cbName' and 'cbData' are both 32-bit integers that allow far more bytes to be reserved for a colour profile.

It's not clear why a stack buffer of fixed size has been used instead of a heap buffer allowing for dynamic size, but the insights gained from the process of parsing the



```

mov     edi, edi
push   ebp
mov     ebp, esp
sub     esp, 250h
mov     eax, ___security_cookie
edx     [ebp+Data]
push   esi
mov     esi, [ebp+arg_0]
mov     [ebp+var_4], eax
push   edi
xor     eax, eax
mov     ecx, 93h
lea     edi, [ebp+var_250]
rep     stosd
and     [ebp+var_244], eax
mov     eax, [ebp+arg_8]
mov     [ebp+var_240], eax
push   edx; lpstring2
lea     eax, [ebp+var_20C]
push   eax; lpstring1
mov     [ebp+var_250], 50534F43h
mov     [ebp+var_24C], 400h
mov     [ebp+var_248], 24Ch
call   ds:lstrcpyW(x,x)

```

Figure 5: Unpatched EMF vulnerability.

EMF structure can now be used for defensive purposes. They allow us to build a generic detection mechanism for MS08-021-specific exploits by checking for EMR_COLORMATCHTOTARGETW records (and probably related ones) having a 'Data' field in relation to a filename with more than 0x104 wide characters.

SOFT-FOCUS EFFECT

With the record's structure in mind and looking at the exploit's hex-dump again (Figure 2), it is evident that 'Data' doesn't contain a valid colour profile name and the number of wide characters in total exceeds 0x104 (the latter not being visible in the screenshot). Instead, the question of where exactly the exploit's shellcode has been placed is answered by disassembling the record's suspicious-looking 'Data' field at file offset 0xB4.

The shellcode (Figure 6) is XOR-encoded by an eight-bit key in order to avoid null-words, which could render the exploit ineffective when interpreted as zero-termination of Unicode strings. Once the shellcode has decoded itself on the stack, it uses the Process Environment Block (PEB) to find the base address of kernel32.dll in order to import several API functions by conducting a hashed string comparison of the APIs. These are used to download and install a backdoor, with a file name as 'unambiguous' as it can get – 'word.gif', from igloofamily.com (a domain hosted in Korea).

The shellcode possesses the ability to bypass modern behaviour blockers that detect API functions being called

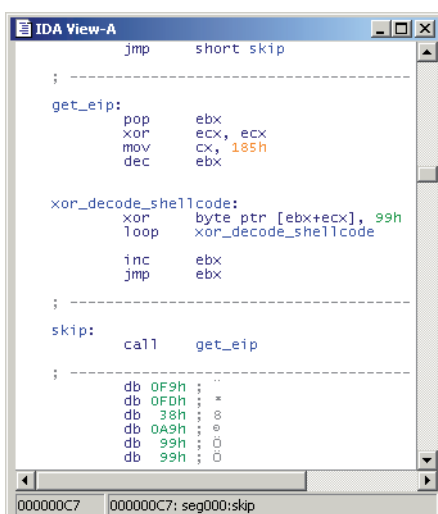


Figure 6: Shellcode found within an exploit's EMR_COLORMATCHTOTARGETW record.

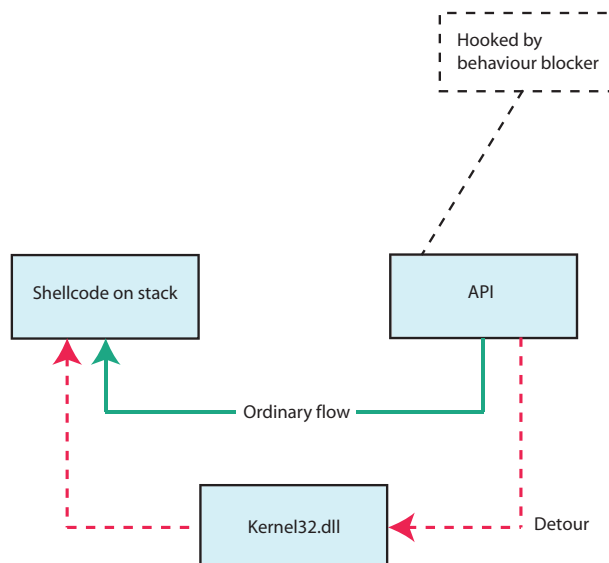


Figure 7: Behaviour blocker evasion mechanism.

from unusual places like the stack. This circumvention works by inserting a faked return address into the stack before calling an API function. The detoured return address points into a 'ret' assembler instruction within kernel32.dll, which in turn pops the real return address off the stack to be able to return control to the real caller. By doing so, a behaviour blocker is tricked into believing the actual caller is within the Windows kernel32 library area instead of shellcode on stack, thus being legitimate (Figure 7). The 'ret' instruction that is used as a detour is searched for manually within the kernel32.dll module near the address of WinExec().

In large part, this downloaded variant of the 'Poisonivy' backdoor consists of multi-layered, encrypted and position-independent code. Position-independent code not only makes it harder to read the disassembly, in this context it is necessary since the code is injected into the memory space of running processes such as 'explorer.exe' or 'msnmsgr.exe' (via WriteProcessMemory and CreateRemoteThread APIs).

As soon as the injected thread is executed, it creates a 'hidden' copy of the trojan as an NTFS alternate data stream (ADS) named 'win_socks.exe', attached to the 'system32' folder in the Windows directory. Its launch upon system reboot is ensured by the creation of an 'Active Setup' registry key in 'HKCU\Software\Microsoft\Active Setup\Installed Components\{E5C1F9EA-A8FE-FCBC-9F3D-C2791859730F}', named 'StubPath'.

Afterwards the code stays in a loop waiting for *MSN Messenger* (*msnmsgr.exe*) to be run in order to inject another piece of self-decrypting code into it. This remote thread will initiate a connection with 'word.4pu.com', 'word1.4pu.com' and 'word2.4pu.com', giving the attacker total control over the compromised system. Since it is a program that is usually allowed to pass the firewall on most systems, *MSN Messenger* makes an ideal target for a backdoor like this variant of 'Poisonivy'.

CONCLUSION

Looking back at its history, the GDI has been a popular target among attackers. Besides the relatively recent EMF holes, GDI has also been vulnerable to remote attacks in the past, like the two ANImated cursor vulnerabilities (MS05-002 and MS07-017), a JPEG vulnerability in GDI+ (MS04-028) and WMF vulnerabilities (MS06-001). But thankfully there has been evolution: it's good to see the quality of the code going through different security stages, from using unsafe functions to using safer functions up to using proprietary safe functions in combination with the /GS compiler option.

As demonstrated by this article, it's not just lucrative business for the bad guys to perform binary code auditing and have a thorough look at the facts, but also for the good guys who need to defend these attacks. Vulnerabilities, like February's *Adobe Reader* vulnerability (CVE-2008-0655) and April's MS08-021 GDI vulnerability, begin to be exploited just a few days after the vendor's patch release – much faster than corporations may need to verify and deploy the patches. If we know the exact causes of vulnerabilities, we do not need to wait for the first exploits to fall into our hands in order to protect and defend ourselves effectively. Instead, we are able to tell apart benign structures from abnormal ones and thus can defend proactively against upcoming attacks.

REFERENCES

- [1] Ferrie, P. Inside the Windows Meta File format. *Virus Bulletin*, February 2006, pp.5–8. <http://www.virusbtn.com/vba/2006/02/vb200602-wmf>.
- [2] Microsoft Security Bulletin MS08-021. <http://www.microsoft.com/technet/security/bulletin/ms08-021.mspx>.
- [3] Enhanced Metafile Format Specification Revision 2.0. <http://msdn2.microsoft.com/en-us/library/cc204166.aspx>.



VB2008 OTTAWA 1–3 OCTOBER 2008

Join the *VB* team in Ottawa, Canada for *the* anti-virus event of the year.

- What:**
- Three full days of presentations by world-leading experts
 - Automated analysis
 - Rootkits
 - Spam & botnet tracking
 - Sample sharing
 - Anti-malware testing
 - Corporate policy
 - Business risk
 - Last-minute technical presentations
 - Networking opportunities
 - Full programme at www.virusbtn.com

Where: The Westin Ottawa, Canada

When: 1–3 October 2008

Price: Special *VB* subscriber price \$1795

**BOOK ONLINE AT
WWW.VIRUSBTN.COM**



FEATURE 2

STRIKE ME DOWN, AND I SHALL BECOME MORE POWERFUL!

Aditya Kapoor and Rachit Mathur
McAfee Avert Labs, USA

In the *Star Wars* film, the character Obi-Wan Kenobi famously says to arch enemy Darth Vader: ‘If you strike me down, I shall become more powerful than you could possibly imagine.’ In a twisted sense of life imitating art, with the roles of good and evil played out in reverse, the rootkit that has been dubbed one of the stealthiest ever seen in the wild is back – with improved defences and new stealth code.

Early variants of Mebroot (aka StealthMBR) appeared in late 2007 and achieved notoriety for their stealth techniques. This article discusses the new variants released since mid-March 2008. These are particularly interesting due to the number of improvements made to this already complex threat. This article adds to previous discussions of the threat [1, 2] and discusses ways to counter the newly introduced challenges.

KNOW YOUR WEAKNESS

In its earlier variants, Mebroot uses advanced infection techniques to modify critical areas in the system. It uses user-mode functions such as CreateFile, WriteFile etc. to access \\.\PhysicalDriveX and write directly into sectors of the disk to infect the MBR, install its own loader code, save a copy of the original MBR and install its kernel-mode driver.

Using the infected MBR, Mebroot gains full control very early during the *Windows* load process, slips its malicious payload into the kernel memory and hooks the IRP_MJ_READ and IRP_MJ_WRITE function pointers of \Driver\Disk. Using these low-level IRP dispatch table hooks it is able to filter out read/write requests to its malicious disk sectors including the infected MBR. No file or registry is required for the malware to survive reboots once it is installed. These techniques pose a serious challenge for detection and repair.

Despite initially seeming difficult to overcome, anti-malware developers were able to identify weak points in the malware’s defences and come up with successful solutions. However, Mebroot’s author(s) seem to have learned from its earlier shortcomings and have made enhancements to the code in the new variants to reinforce the malware’s defences and escalate the arms race.

```

Disassembly
Offset: 8196687e
81966864 893554279d81 mov     dword ptr ds:[819D2754h],esi
8196686a ff15ecd9c81  call   dword ptr ds:[819CFDECh]
81966870 ff75ec      push  dword ptr [ebp-14h]
81966873 e8ccdeffff  call   81964744
81966878 5f          pop     edi
81966879 5e          pop     esi
8196687a 5b          pop     ebx
8196687b c9          leave
8196687c c3          ret
8196687d cc          int     3
8196687e ff2524279d81 jmp     dword ptr ds:[819D2724h] ClassCreateClose
81966884 ff255c279d81 jmp     dword ptr ds:[819D275Ch] ClassInternalIoControl
8196688a ff2564279d81 jmp     dword ptr ds:[819D2764h] ClassDeviceControlDispatch
81966890 ff2520279d81 jmp     dword ptr ds:[819D2720h] ClassShutdownFlush
81966896 ff252c279d81 jmp     dword ptr ds:[819D272Ch] ClassDispatchPnp
8196689c ff2528279d81 jmp     dword ptr ds:[819D2728h] ClassDispatchPower
819668a2 ff2560279d81 jmp     dword ptr ds:[819D2760h] ClassSystemControl
819668a8 85db       test   ebx,ebx
819668aa 56          push  esi
819668ab 0f84b0000000 je     81966961
    
```

Figure 1: Dummy code hook table.

The following two sections identify the weaknesses in the original code and the measures taken by the malware’s authors(s) to address them. Thereafter we discuss how these strengthened defences can still be broken.

REINFORCE THOSE DEFENCES

Many heuristic detection tools search for inconsistencies to raise an alert. Earlier variants of Mebroot hook only two out of 11 valid pointers in the dispatch table of \Driver\Disk, resulting in inconsistent pointer target locations and thus raising suspicion (normally all pointers of \Driver\Disk point to the ClassPNP module).

To avoid suspicion in this scenario the author(s) of Mebroot created a dummy hook table within its code and hooked all other valid IRP_MJ_* functions in addition to IRP_MJ_READ/WRITE, so that they all point within the same module and thus do not raise any suspicion. Since the malware does not really want to intercept calls other than to read/write functions, all the other pointers point directly to one of a series of jumps in the rootkit module referred to as a dummy hook table. As shown in Figure 1, these jumps relay control immediately to the original routines in ClassPNP.

```

b8cd12aa9f  mov     eax,offset CLASSPNP!ClassCreateClose
894338      mov     [ebx+38h],eax
894340      mov     [ebx+40h],eax
b8aebaa9f9  mov     eax,offset CLASSPNP!ClassReadWrite
894344      mov     [ebx+44h],eax
894348      mov     [ebx+48h],eax
b84cc3a9f9  mov     eax,offset CLASSPNP!ClassShutdownFlush
c74374acfaa9f9  mov     [ebx+74h],offset CLASSPNP!ClassInternalIoControl
c74370d4c3a9f9  mov     [ebx+70h],offset CLASSPNP!ClassDeviceControlDispatch
894378      mov     [ebx+78h],eax
89435c      mov     [ebx+5Ch],eax
c783a4000000d14aa9f9  mov     [ebx+0A4h],offset CLASSPNP!ClassDispatchPnp
c783900000008fd0ba9f9  mov     [ebx+90h],offset CLASSPNP!ClassDispatchPower
c78394000000611eaaf9  mov     [ebx+94h],offset CLASSPNP!ClassSystemControl
39bec8000000      cmp     [esi+0C8h],edi
0f8517170000      jne    CLASSPNP!ClassInitialize+0x1a
    
```

Figure 2: The ClassInitialize function of ClassPNP.sys.

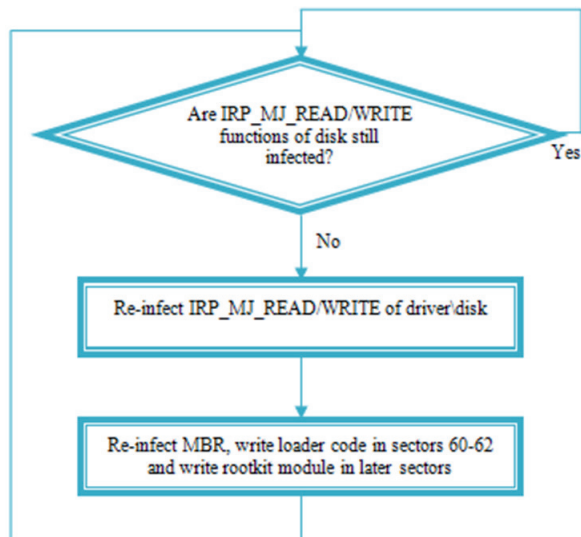


Figure 3: Watcher thread self-preservation logic.

Another way in which an anti-malware tool can detect rootkits in memory is by comparing various kernel structures to find inconsistencies. There are a couple of places in the kernel where the original pointers of the IRP dispatch table of `\Driver\Disk` are normally saved, which a detection tool can use to find the inconsistency. One is the IRP dispatch table of `\Driver\CDRom` (which has exactly the same pointers as `\Driver\Disk`) and the other is the `ClassInitialize` function in the `ClassPNP.sys` file, which is an exported function and stores the original IRP pointers for its internal use at a certain offset (see Figure 2).

To bolster its defences against such an attempt at reading the original pointers and restoring them, the new variants of Mebroot hook the IRP dispatch table of `\Disk\CDRom` and change the pointers to match the location of corresponding hooked dispatch routines of `\Driver\Disk`. Similarly, Mebroot also patches all the original pointers as listed in the `ClassInitialize` function of `ClassPNP` to match its hooks (shown in red in Figure 2).

Finally, the rootkit uses a watcher thread as a fail-safe method to prevent itself from being removed even if an anti-malware program is able to ascertain the original IRP read/write pointers.

As shown in Figure 3, the watcher thread watches continuously for any attempt to restore the original IRP read/write hooks. As soon as these hooks are modified the thread does four things in the following order:

1. Re-sets the `IRP_MJ_READ` and `IRP_MJ_WRITE` pointers to its own filtering routine.

2. Attempts to rewrite the MBR at sector 0.
3. Attempts to rewrite the rootkit loader code and original MBR code at sectors 60, 61 and 62.
4. Attempts to rewrite the rootkit module in the later sectors of the disk.

A more detailed description and annotated code of watcher thread can be found at [3].

TAKING A CLOSER LOOK

From the perspective of detection and repair one of the most interesting elements for analysis is the rootkit's filtering routine for read/write. The filtering routine maintains a memory image, 'Fake Sect', of the first 63 sectors as they would have appeared on the clean machine. Now any read/write requests for the first 63 sectors are redirected to this 'Fake Sect' memory buffer instead of actually reading from or writing to the disk. This not only gives a false impression that the MBR is clean, but also makes it seem as if write operations are working (it would raise suspicion if write operations in these sectors didn't work, but the 'Fake Sect' ensures that this illusion persists).

As shown in Figure 3, the IRP read/write pointers are re-infected before the sectors, the result of which is that these sector write requests also have to go through the malware's own filtering routine. To allow its own requests to actually write to the disk and not to 'Fake Sect', the filter function checks whether a magic seed is present at offset `0x40` of IRP [2], as shown in Figure 4. These special IRPs cannot be constructed using a user-mode call, which is why the new variants use the `IoBuildSynchronousFsdRequest` API to construct an IRP with `IRP_MJ_WRITE` as the major function, then modify it and write directly into various sectors of the disk by using the device object of the disk in the call to `IofCallDriver`. Figure 5 shows the rootkit function used to create and send the IRP.

In the variants that we analysed, the watcher thread did not exactly work as intended, and steps 2, 3 and 4 mentioned in the previous section failed; the instruction for `mov` to `0x40` offset in IRP shown in Figure 5 does not get executed. So, for steps 2, 3 and 4 the thread function is not able to construct an IRP with the magic seed at `0x40`. Thus the

Disassembly			
Offset: 8196150e			
819614df	7446	je	81961527
819614e1	8b4640	mov	eax, dword ptr [esi+40h] -- PIRP + 0x40
819614e4	3b05101f9d81	cmp	eax, dword ptr ds:[819D1F10h -- cmp with magic seed
819614ea	742e	je	8196151a

Figure 4: Code snippet from filter function.

```

push [ebp+arg_8] ; Event
movzx eax, [ebp+arg_4] ; StartingOffset
push [ebp+arg_C] ; Length
push [ebp+arg_10] ; Buffer
push [ebp+arg_0] ; DeviceObject
push eax ; MajorFunction
call nt!IoBuildSynchronousFsdRequest ; to build IRP
cmp eax, esi
jnz short loc_443A40
mov eax, 0C0000001h
jmp short loc_443A74
-----
mov ecx, [ebp+arg_14] ; CODE XREF: sub_4439E6+51fj
cmp ecx, esi
jz short loc_443A4A ; DeviceObject
mov [eax+40h], ecx ; Mov to PIRP + 0x40h. Try to modify IRP.
mov ecx, [ebp+arg_0] ; DeviceObject
mov edx, eax ; PIRP (pointer to IRP)
call nt!IoCallDriver ; Send IRP

```

Figure 5: Code snippet to construct and send IRP.

malware gets caught in its own trap and its own read/write requests are also redirected to the 'Fake Sect' and not to the actual disk.

Due to this bug, when an attempt is made to restore the IRP table, the watcher thread ends up infecting the 'Fake Sect' that was supposed to present a clean view to tools accessing these sectors. Now, anti-malware tools trying to scan the MBR may start triggering infected MBR detections based on this 'Fake Sect', but any repair attempts that seem to work will actually only be reflected in 'Fake Sect' and the disk will remain infected. One should use caution while testing solutions with these buggy variants.

In the following discussion we assume that the watcher thread functions properly and writes on the disk instead of 'Fake Sect'.

BUT DEFENCES ARE MEANT TO BE BROKEN

Detection of this threat based on MBR scanning once the rootkit is active is challenging, but in-memory detection of the rootkit can easily be achieved. One of the popular methods of cleaning before these new variants arrived was to make use of the original IRP_MJ_READ/WRITE addresses of \Driver\Disk from ClassPNP!ClassInitialize, but as discussed earlier, that is no longer possible. Additionally, cleaning in the new variants involves not only restoring the MBR but also making sure that the MBR is not re-written by the watcher thread. So, to clean this threat we first have to deactivate the watcher thread before cleaning the MBR.

Deactivation of the watcher thread can be achieved simply by suspending it or by patching the watcher thread code to force self-termination or by inserting NOP instructions in the chunks of code where it checks and writes back IRP

hooks and where the MBR is modified. Of course one has to be careful when patching code or suspending threads. Once the thread has been deactivated the MBR can be restored using normal user-mode methods to write to the MBR. However, as in the previous variants the IRP needs to be restored as well. Restoring the IRP hooks can still be a challenge even after the watcher thread has been deactivated because the original function pointers are difficult to ascertain, especially due to the CD-ROM hooks and the patched ClassInitialize function. But there are still other ways to restore the IRP hooks after the thread is deactivated.

As another option one could patch the magic seed value to zero, causing the filter function to allow all read/write requests to pass through except for the watcher thread's own requests. Once this is done, normal MBR detection and repair is sufficient. This approach patches data instead of code, which is safer and does not require thread deactivation or IRP restoration.

Once the detection flag has been raised, the good old manual repair method with booting from an external medium is always a good fall-back solution.

CONCLUSION

Motivated by profit, the author(s) behind this threat have shown that they have the ability to take offensive technology concepts [4] and convert them into real-world malware. They have also identified the measures being taken to remediate the threat and have come up with a strengthened wave of variants. This new wave of variants with all its changes has presented challenges for a lot of anti-malware developers. On the brighter side, whatever the complexity of these variants, anti-malware products have been able to react quickly in providing successful detection and cleaning.

REFERENCES

- [1] GMER. Stealth MBR rootkit. <http://www2.gmer.net/mbr/>.
- [2] Florio, E.; Kasslin, L. Your computer is now stoned (...again!). Virus Bulletin, April 2008, pp.4–8. <http://www.virusbtn.com/vba/2008/04/vb200804-MBR-rootkit>.
- [3] Kapoor, A.; Mathur, R. Exploring StealthMBR Defenses. <http://www.avertlabs.com/research/blog/index.php/2008/03/23/exploring-stealthmbr-defenses/>.
- [4] eEye BootRoot. <http://research.eeye.com/html/tools/RT20060801-7.html>.

FEATURE 3

SYSTEM CLEANING: GETTING RID OF MALWARE FROM INFECTED PCS

Maik Morgenstern & Andreas Marx
AV-Test.org, Germany

Malware has evolved quite significantly in the decades it has been around. In the beginning, file-infesting viruses were the main threat (*although there were times in the very early days*



when boot-sector viruses caused the most infections - Ed). Simple at first, they quickly evolved into more complex incarnations, using techniques such as self-encryption and eventually leading to polymorphic variants.

The anti-virus industry's response was also pretty simple at first. AV products were able to detect a virus and tell the user about it, but no cleaning routines were provided. Infected files had to be replaced with a clean version of the original in order to fix the problem. However, with the increasing complexity of operating systems and the ability of users to install more and more applications this soon became impractical. In response, AV vendors introduced disinfection capabilities to their products, which had to deal with more complex virus creations from year to year, covering not only executable files but also *Office* documents and other file types.

However, malware evolution did not stop at that point. Rather than simply infecting files, multi-component approaches affecting many parts of the system became the standard in malware and remain so today. This is especially true in the case of spyware, which traditionally makes a lot of changes to the file system as well as to the registry. Many other malware attacks also consist of several components that have to be dealt with by AV software. The easy cases with only one process, one file and one registry value are certainly getting rarer and the complexity of malware threats is increasing. This refers not only to the magnitude of changes to the system, but also to the techniques used and the overall behaviour of the malware. Rootkit techniques and anti-removal measures are some of the most challenging for detection software [1].

The AV industry responded to this new challenge and learned to remove the malicious components from the system. Simple cases were easy to deal with: terminate the

process, remove the executable and maybe even handle corresponding registry entries. However, with the increase in volume and complexity of malware the removal of malicious components became more difficult. In order to remove a malicious item successfully from a system, it is necessary to know exactly what to remove. This means that some kind of disinfection routine must be in place, which in turn requires some analysis of the malware. This pretty much describes the way in which AV vendors traditionally did the job: AV researchers analysed the malware, identified the changes to the system and could provide a disinfection routine with the next update. While this approach worked well some years back with smaller volumes of slower-spreading malware, it has serious drawbacks now. Often, it just takes too long for a dedicated disinfection routine to become available.

The solution seems obvious: generic approaches for disinfection, which don't rely on an analysis from the AV vendor. While there are certainly promising attempts that can handle the simpler cases, the more complex cases still pose a problem. The components that are detected by static or dynamic mechanisms can usually be removed, however this is not always true for linked components, be they files and directories or registry entries. This means that some parts of the malware can indeed be successfully removed, but others which can still be a threat to the system remain. These findings and more details are available in [2].

With the above in mind it is clear that the testing of system cleaning capabilities is still a very valuable exercise. There are many variables that have to be considered by the AV vendors which could prevent successful cleaning. It is useful, therefore, to run tests that determine how well today's products are able to handle system disinfection and how well they can cope with special circumstances such as anti-removal techniques. We will describe the basic requirements of such tests, present some of the details of our testing procedures and look at the results of some of our recent tests.

SAMPLE SELECTION AND CREATING THE TEST SET

As for most tests, sample selection is one of the first and most important steps in the testing process. A wide variety and a large number of samples must be used in order for the results to have statistical relevance. Due to the complex nature of the tests, the test set cannot be as large as it would be for a static scan test, but other factors can still help ensure its relevance.

The basic requirement is that the samples are active and actually perform changes to the system. The likelihood of the products being able to detect the samples must also

be considered, because this will influence the disinfection process. There may be signature-based detection which could trigger a dedicated disinfection routine, proactive detection which might lead to a generic disinfection routine, or no detection, which obviously won't trigger any disinfection process.

Besides these basic requirements, different malware types and families should be chosen for the test set, to cover different behaviour and levels of complexity. The samples should also be currently spreading in the wild, to reflect real-world threats. Finally, the sample selection and analysis process must be performed on the same operating system and under the same conditions as those in which the test will be carried out. This is necessary to make sure the criteria that have been used for selecting the samples still apply when testing.

The tester needs to know exactly what changes to the system are performed by the malware. In order to determine this, an automated analysis tool is used, which records every change to the file system and registry and discovers newly created processes. This gives a comprehensive overview of the relevant malicious activities on the system and helps in the sample selection process.

The same tool can also be used to solve two common problems encountered when testing active malware: reproducibility and comparability. Since active malware may change its behaviour depending on several variables, including some that cannot be controlled by the tester, the actions of the malware – and therefore the changes to the system – may be different on every test run. This could prevent the tester from reproducing a test result, since the malware may never act as it did before. It could also prevent the tester from comparing the cleaning performance of one product against that of another, since they might have to cope with different malware behaviour and some may be easier and some harder to deal with. These issues are particularly likely to arise with malware which downloads additional components from the Internet.

Since the scope of this test extends only to the cleaning of an infected system and not the prevention of infection, the analysis tool can be used to help overcome these problems. The recorded system changes are saved in a special archive format (packages) which can be used to restore the whole infected state on any system at any time. This easily solves the two problems mentioned above: the package can be used to reproduce exactly the same infected system state as often as necessary. This in turn means that exactly the same conditions can be created for every product in the test and their cleaning performance can easily be compared.

PERFORMING THE TEST

The testing procedure is straightforward, especially with the help of the packages from the analysis tool. We use an image with an up-to-date installation of the AV product under test and turn off the on-access protection in order to be able to restore the infected system state. This is done by replaying the system changes recorded in the package. After this is finished and the system is in a known infected state a system scan is carried out using the default options. Whenever anything is detected, we let the AV product run its cleaning or disinfection routines to remove the malicious components. After allowing any required reboots and additional scanning and cleaning steps, the final system state is determined using the same tool as used in the analysis and preparation steps. Since we know exactly which changes to the system have been made by the malware, we can also determine exactly which components have been removed by the AV software and which components have been left behind.

This gives us the raw information as to what and how much has been detected and removed, but it does not represent the cleaning success. In order to assess this, the system changes must be categorized by risk level.

First, there are the changes that are clearly malicious, which must be removed, reverted or set to default settings. These include malicious executables and the linked start entries in the registry or file system, but also extend to modifications to the hosts file as well as altered security and browser settings in the registry.

The second category contains unpleasant or unwanted, but not actually dangerous, system changes. One example is pornographic images that accompany a lot of malware these days. This should certainly be handled by the AV product in corporate environments and home users will want them removed too, especially where children use the computer.

The last category contains changes that don't have any real effect but are visible on the system. These include directories, trash or 0-byte files or junk registry entries that are not used by the operating system.

In order to clean a system successfully, the bare minimum an AV product must be able to do is to handle the first category of changes and disable the malware effectively. This means the malicious processes must be terminated, the corresponding files and the start entries must be removed. Any changes to security and browser settings as well as modifications to the hosts file should at least be detected and reported to the user. Since the pre-infection settings are often unknown, it is not possible simply to reverse these changes, but reverting to the

Product	Version	Detection of inactive samples	Detection of active malware	Disabling of active malware	Removal of active malware
	Reference	5	5	5	5
Avira Antivir PersonalEdition Classic	7.06.00.270	5	4	4	3
BitDefender Antivirus 2008	11.0.0.15	5	4	4	2
BullGuard Internet Security 2008	8.0.0.1	5	4	4	2
F-Secure Anti-Virus 2008	8.00 build 101	5	4	4	2
G DATA AntiVirus 2008	18.3.7338.740	5	3	3	3
Kaspersky Anti-Virus 7.0	7.0.0.119	5	5	5	4
McAfee VirusScan Plus 2008	12.0 Build 176	5	4	4	3
Symantec Norton AntiVirus 2008	15.0.0.58	5	5	5	4
Panda AntiVirus 2008	3.00.00	5	4	4	3
Windows Live OneCare 2	2.0.2500.14	5	4	4	3

default settings is always an option that can be offered by the AV software.

What we often see is that only the malicious executables are handled. Additional dropped files, registry entries and other changes are not dealt with. This is critical for several reasons. The first has been explained above – many changes are themselves dangerous, e.g. in the case of changed browser settings, the user might be redirected to a malicious website that will infect the system with the latest version of the malware again. Another reason is the uncertainty in which the user is left when not all relevant components of the malware are removed. Especially in the case of an infection, a user might want to obtain a second opinion. This could lead to the detection of the left-over malware components by a second AV product and the user will most likely lose confidence in his original security software. The increasingly common ‘light grey’ software products that pose as security software but actually produce rather strange outputs may compound this problem [3]. These applications do not have any real eligibility to be on the market, but ‘detecting’ the left-over components from an incomplete system disinfection might just be what they were looking for as justification.

Besides handling the first category of changes, it would of course be very desirable to handle the other categories as well. Not doing this will not usually mean a failure in the test – as long as the malware is effectively disabled – but the product’s failure to deal with all system changes will be reported.

SOME TEST RESULTS

In this section we will present a few small-scale test results, which illustrate some of the common problems encountered but also show that some products are able to handle the system cleaning task successfully. These results have been published in the German *ComputerBild* magazine [4].

The test was carried out at the beginning of 2008 on *Windows XP* (32-bit, SP2) and the products (in their most current versions) were updated and then frozen on 7 January 2008. The test was carried out as described above.

The results presented here are from tests run against five samples taken from the then current WildList – meaning that signature-based detection of the original sample should be guaranteed. There were three rather easy ones: Win32/Rbot!FB26, Win32/Spybot!ITW203 and Win32/Stration!69F2, as well as Win32/Feeds!8897, which uses rootkit techniques, and Win32/Rontokbro!E517, which tries to terminate AV software. The behaviour of the latter two samples complicated the cleaning process for some of the products.

While all products were able to detect the malware samples in an inactive state, there were some problems when they were already installed and active on the system. *G DATA* and *BullGuard* failed to detect the Win32/Feeds!8897 infection due to its use of rootkit technologies and were consequently not able to clean the system. All the others were able to detect and disable

this threat, however only *Kaspersky* and *Norton* achieved full removal. The remaining products didn't handle the 'ShellServiceObjectDelayLoad' registry entry that was used to restart the malware on reboot and could possibly cause false positives if not removed.

The other problematic sample was Win32/Rontokbro!E517, which terminated seven out of the ten tested AV products or prevented them from scanning. Only *BullGuard*, *Kaspersky* and *Norton* were able to deal with the sample and disable it. However, there were still some problems. The malware disabled the editing of the registry with the 'DisableRegistryTools' entry and none of the products dealt with this. While it is perfectly understandable for this entry not to simply be set back to the default value – which would allow editing of the registry again and may be different from the pre-infection state – it is not clear why this change was not reported to the user. An analysis of the sample in the lab certainly detected the change and it is also safe to assume that most users do not prevent access to their registry. This makes it pretty clear that the disabled registry would in most cases be the result of the malware behaviour and should therefore be reported.

Another issue was the modified hosts file. The *Norton* product did clean some parts of it, especially those that affected *Symantec* addresses, but it left a lot of other bad entries. The other two products that were able to handle this sample simply moved the file into the quarantine. While this effectively disables the malicious intent, it also removes user entries that may be necessary for the system to work as expected.

The other samples didn't pose any serious problems to the AV products: the Win32/Rbot!FB36 sample challenged *BitDefender*, *BullGuard* and *F-Secure* a little with its run registry entry that was left behind by these products, but Win32/Spybot!ITW203 and Win32/Stration!69F2 were both handled effectively by all products.

CONCLUSION

Preventing an infection when the malware sample is known is rather easy. Heuristic and generic detection as well as behaviour-based approaches are a big help in detecting unknown malware and preventing an infection. However, none of these approaches is 100% safe, and there is always the chance that new malware will remain undetected and infect systems. Also, we are well aware that some users do not use up-to-date AV software and only wake up when it is too late and discover an infection on their system. Then is the time for system cleaning routines.

As we have pointed out above, there are cases where AV products work perfectly well, not only disabling the threat

but also removing all relevant parts. This is possible when a dedicated disinfection routine is available or if it is an easy case that can be handled by a generic routine. However, not every piece of malware is simple, and when a more complex piece of malware is encountered – such as one that tries to evade detection and removal and which clutters the system with lots of different components – some AV products show certain weaknesses.

The problems may even start with detection of the malware, because some products cannot handle rootkit techniques or because the malware terminates the security software. But even when it is detected, this does not mean that all parts of the malware will be disabled.

Finally, there is the removal of the malicious components – the performance of many current AV solutions in this area is disappointing in many ways. Registry entries are not handled or only some of them are removed, security and browser settings are ignored and the hosts file is only partially cleaned or simply quarantined. System cleaning involves a lot more than just detecting the malware process and removing the corresponding file. Depending on the complexity of the malware, many more steps might be necessary and must be taken carefully.

In order to solve some of the problems, there is always the option of using a bootable rescue media. Since the malware (and a possible included rootkit) is not active then, no scanner can be terminated. However, this does not replace the need for thorough analysis of current threats and the further development of better generic disinfection routines. Both of these are needed not only to disable (parts of) the malware, but also to remove all relevant components, to keep the user in a safe and confident state.

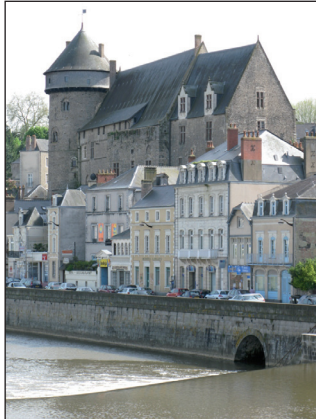
REFERENCES

- [1] Bruce, J. The challenge of detecting and removing installed threats. Proceedings of the 16th Virus Bulletin International Conference, pp.61–64. 2006.
- [2] Brosch, T.; Morgenstern, M. Malware removal – beyond content and context scanning. Proceedings of the 17th Virus Bulletin International Conference, pp.211–217. 2007.
- [3] Schouwenberg, R. The (correct) detection of light grey software. Proceedings of the 16th Virus Bulletin International Conference, pp.52–55. 2006.
- [4] Pursche, O.; Otten, M. Abserviren (to polish off malware). ComputerBild 06/2008, pp.60–67. <http://www.computerbild.de/>.

CONFERENCE REPORT

EICAR 2008, C'ÉTAIT MERVEILLEUX!

Eddy Willems
Kaspersky Lab and EICAR, Belgium



This year EICAR held its 17th annual conference at the conference centre Les Ondines in Laval, France. After financial problems forced the organizers to cancel the conference in 2007, it was a brave decision to go ahead and plan for an event in 2008. Laval may seem an unusual choice for the location of an international conference in its come-back year, but

the excellent conference facilities on offer at Les Ondines were enough to offset any disadvantage of the venue being off the beaten track.

This year's conference theme was 'IT security is facing a paradigm shift – new threats and more subtle methods of attack require different approaches and solutions'. The theme draws attention to the issues arising from the reality of an 'anytime, anywhere' web and an increasingly invisible enemy.

The conference was opened by Professor Dr Nikolaus Forgo from the University of Hannover and Vienna, who gave a keynote speech about prosecution and law enforcement in the context of IT security solutions development. Professor Forgo will lead a new legal advisory board that EICAR is setting up in response to the increasing role of legal issues in the context of IT.

Professor Forgo's presentation made it clear that even gathering information from a network is not as easy, from a legal point of view, as most of us believe. I think that many of us in the audience would think twice even about using some of our sniffer tools again.

Next, two of Professor Forgo's students presented a deep look at the criminalization of hacker tools in the new German law and compared it with other European legal systems.

After the best conference lunch I've had in my 17 years of attending conferences, the real agenda kicked off with presentations split between an industry track and an academic track.

François Paget from *McAfee* presented an interesting view of the malware problems related to virtual worlds such as *WoW* and *Second Life*. Meanwhile, in the other track Vanja Svajcer and Boris Lau of *Sophos* looked into virtual machine detection in malware using a dynamic-static tracing system. Richard Ford outlined a danger theory-based artificial immune system for the MANET (mobile ad hoc networks) environment. He showed how a simple reputation system can be improved in this environment by considering the experiences of similar systems.

Next up, 'Simulating malware with MAISim', presented by Rafal Leszczyna, Igor Nai Fovino and Marcelo Masera from the European Commission, was quite a controversial talk about a mobile agent framework used to address security assessments based on simulation of attacks against the systems – something like a combination of penetration and malware testing on real systems. In my opinion these researchers should exercise caution as their work treads a precarious line and could easily be misinterpreted or misused. Fraser Howard of *Sophos* rounded off the first day's sessions with a nice, deep overview of Web 2.0-based attacks.

The gala dinner that evening will be remembered by most of the attendees as being tastier even than the aforementioned lunch – you simply can't beat the real France for good food and wine! The dinner was held at the old Laval castle, a magnificent mediaeval palace in the centre of the picturesque town.

The second day of the conference opened with a realistic and deep view of Win32.Ntldrbot (Rustock.C) given by Boris Sharov of *Doctor Web*. Afterwards, Mario Vuksan from *Bit9* described the use of a whitelisting approach to improve the quality of security software and effect a radical transformation of anti-malware and HIPS products. He also demonstrated the power of this system in tracking down new types of malicious software. Finally, a talk by Andrei Gherman of *Avira* about the latest botnet trends gave a very good view of the real problem and described several different monitoring solutions.

With more than 26 papers and talks it is not possible to provide a summary of them all, so I urge readers to look them up on the EICAR website: <http://www.eicar.org/conference/>.

This year's EICAR conference was a fresh start and I firmly believe that EICAR is running once again in the right direction with some interesting new projects on the horizon. The quality of the conference papers was excellent – just ask any of the people who attended. Let's hope for some even more interesting presentations next year, when the conference moves to Germany, taking place either in Dresden or in Berlin.

COMPARATIVE REVIEW

UBUNTU LINUX 8.04LTS SERVER EDITION

John Hawes

Once again the VB100 review rolls around to its annual visit to a *Linux* platform, and once again the same questions arise. The ever-growing hordes of *Linux* and open-source aficionados continue to revel in the relative impenetrability of their security model and in the scant attention paid to them by malware creators. Why, I am asked on what seems like a daily basis, would I want to run anti-virus on my *Linux* box? What have I to fear? A tiny handful of malware with puny penetration levels is surely a risk worth taking, runs the standard argument.

Of course, this is quite beside the point; while *Linux* as a desktop operating system continues to nurture its small, but generally keen and committed user base, it is when running on a server that it is really at home, continuing to dominate at gateways and scattered throughout corporate and academic networks. Fileserver systems, with their arrays of *Windows* clients storing and transferring all manner of things thanks to the delights of *Samba*, can be nasty breeding grounds for network-wide malware infestations if they are not properly protected. However, they can also be used as effective blockades, preventing malicious code from being passed to new targets. Even where desktops feature their own anti-malware systems, corporate policies often (rightly) insist on thorough regimes of protection with no system allowed to operate without malware scanning, no matter how secure it may seem. For this reason, the *Linux* VB100 generates a great deal of interest from our readers in the corporate world, who are not above demanding tests on far more esoteric platforms.

Ubuntu Linux is a relative newcomer to the scene compared to the likes of *SUSE* and *Red Hat*, the even more venerable *Slackware* and, of course, *Debian*, from which *Ubuntu* evolved. The distribution's first release was in 2004, and since then it has seen massive growth, with strong financial backing through its links to *Canonical Ltd* and its entrepreneur founder Mark Shuttleworth, the first African to venture into space.

Ubuntu has shied away from the duality of commercial and free versions adopted by other big-money distros, and embraced the open-source philosophy wholeheartedly. Accompanied by numerous offshoot projects focusing on specific desktops systems and user groups, *Ubuntu's* focus on friendly usability, stability and consistent updating has brought strong penetration of desktops – a poll held last summer found over 30% of respondents were using it,

with its nearest rival, *OpenSUSE*, at 19% and *Debian* at 11%. At the server level fewer details are available, but the server edition seemed more appropriate to our purposes; of course this selection brought with it the likelihood of some compromises in usability, with any cuddly ease of use likely to have been stripped away in favour of efficiency, security and robustness.

The challenge of learning a new platform should, I hoped, be somewhat mitigated by the fairly small number of entries this month – a mere 15 products providing a relatively easy ride between the mammoth *Vista* test last time around and what is likely to be an even more gargantuan array of products in the *XP* test scheduled to take place later in the summer. Looking forward to simple command-line interfaces providing easy access to configuration options, I burned the install image, dusted off my rather neglected *Linux* skills, and ventured bravely into the lab.

PLATFORM AND TEST SETS

Installation of *Ubuntu* was a pretty straightforward process, guided by a pleasant graphical setup process. As usual in VB100 testing I tried to keep things as simple as possible, sticking to the default settings to get as close as possible to an out-of-the-box setup. Of course this policy couldn't be applied perfectly, with some stages such as disk partitioning requiring specific tuning for my needs, but the final setup provided the basic *Ubuntu* fileserver. As expected, this didn't include a desktop environment, so those products with attractive interfaces would not have their full range of offerings investigated, but the command-line style is generally preferred at the server level anyway, with a minimum of 'magic' going on to open potential security holes and drain resources.

Less expected was the absence of other useful items, including an NFS implementation, but a little investigation showed that a large range of extra goodies were available on the install CD. Beyond a few basic steps such as configuring networking, connecting to the lab servers and client systems and copying test samples to the local machines, very little further work was required before taking snapshot images and getting down to business. In the previous *Linux* test (see *VB*, April 2007, p.11) a copy of *dazuko*, the open-source file-hooking software used by many *Linux* products, was prepared on the test machines in advance, but in this case one of the submissions had thoughtfully included a pre-built binary so this step was not necessary (although I had little doubt that some compilation would eventually be required).

Client systems were also prepared, using a standard *Windows XP SP2* image with the *Samba* shares of the test

On-demand detection rates	WildList viruses		Worms & bots		File infector viruses		Polymorphic viruses		Linux samples		Legacy samples		Clean sets	
	Missed	%	Missed	%	Missed	%	Missed	%	Missed	%	Missed	%	FP	Susp.
Alwil avast!	0	100.00%	0	100.00%	0	100.00%	319	87.13%	2	96.67%	1027	97.02%	2	
AVG Anti-Virus	0	100.00%	1	99.94%	7	98.43%	691	73.89%	3	88.33%	710	95.83%		
Avira AntiVir	0	100.00%	0	100.00%	0	100.00%	0	100.00%	6	66.67%	0	100.00%		
BitDefender Security	0	100.00%	0	100.00%	2	98.95%	0	100.00%	4	93.33%	9	99.93%		
Doctor Web Dr.Web	16	97.55%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	3	12
ESET Security	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Frisk F-PROT	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	1	
F-Secure Linux Security	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	
MicroWorld eScan	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	3
Norman Virus Control	0	100.00%	0	100.00%	7	99.15%	765	73.47%	0	100.00%	269	99.00%		
Quick Heal	0	100.00%	1	99.87%	9	98.43%	808	83.86%	7	66.67%	1127	93.95%	2	
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	7	65.00%	0	100.00%		
Symantec AntiVirus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
VirusBuster SambaShield	0	100.00%	2	99.91%	8	99.21%	224	79.29%	6	83.33%	20	99.92%		

servers mapped, with all on-access tests planned to be run from these. To avoid unfairness, network activity was kept to a minimum during the tests, which were run one at a time to further reduce the possibility of unequal treatment.

The test sets were aligned with the March 2008 WildList, which was released a few weeks prior to the test set deadline of 2 May and the product submission deadline of 5 May. The latest WildList included a fairly large number of new additions, but these were concentrated in a few families – most notably a large swathe of W32/OnlineGames trojans, showing further evolution of the WildList into the cybercrime-ridden modern world. Quite a few older items fell from the list, including several strains of W32/Mytob and W32/MyDoom, and also the veteran W32/Nimda, which finally dropped off the list after a marathon stint of officially being in the wild.

My attendance of numerous meetings and conferences this month hampered any efforts to expand the other test sets by more than a minimal amount, with only a handful of items added to the clean and infected sets and the meagre set of *Linux* samples dusted off. The most significant addition was the insertion of a set of *Linux* files into the clean set, to form an extra part of the speed measurements. This time the samples were taken from a separate system from those running the tests, one which had been in heavy use for

some time and thus held a more eclectic range of items. The entire contents of */bin*, */sbin*, */etc* and */opt* were included, making the new set on a par with the others in terms of size on disk, but considerably ahead of them in the number of files it contained.

Finally, the standard archive test set consisted of the EICAR test file embedded in a selection of archive types at a range of depths. These would, as usual, be scanned with the default settings and with ‘all files’ and ‘scan archives’ settings enabled where possible, and detection at a depth of five or more levels on at least half of the set would be considered adequate for a product’s inclusion in the full archive graphs. With everything ready to go, it was time to see how the selection of products fared.

Alwil avast! for Linux 3.1.0

ItW	100.00%	Polymorphic	87.13%
ItW (o/a)	100.00%	Linux	96.67%
Worms & bots	100.00%	Legacy	97.02%
File infectors	100.00%	False positives	2

Alwil’s product arrived as several archive files, which when unpacked were found to contain simple installer scripts which did all the work of setting things up very

On-access detection rates	WildList viruses		Worms & bots		File infector viruses		Polymorphic viruses		Linux samples		Legacy samples		Clean sets	
	Missed	%	Missed	%	Missed	%	Missed	%	Missed	%	Missed	%	FP	Susp.
Alwil avast!	0	100.00%	0	100.00%	0	100.00%	319	87.13%	2	96.67%	1027	97.02%	2	
AVG Anti-Virus	0	100.00%	1	99.94%	7	98.43%	691	73.89%	6	71.67%	710	95.83%		
Avira AntiVir	0	100.00%	0	100.00%	0	100.00%	0	100.00%	2	93.33%	0	100.00%		
BitDefender Security	0	100.00%	0	100.00%	2	98.95%	0	100.00%	4	93.33%	9	99.93%		
Doctor Web Dr.Web	16	97.55%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	3	12
ESET Security	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
Frisk F-PROT	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	1	
F-Secure Linux Security	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	
MicroWorld eScan	0	100.00%	0	100.00%	0	100.00%	1	99.88%	0	100.00%	0	100.00%	1	3
Norman Virus Control	0	100.00%	0	100.00%	7	99.15%	916	66.94%	6	66.67%	269	99.00%		
Quick Heal	0	100.00%	1	99.87%	9	98.43%	808	83.86%	7	66.67%	1173	93.00%	2	
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	8	99.95%		
Symantec AntiVirus	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%	0	100.00%		
VirusBuster SambaShield	0	100.00%	2	99.91%	8	99.21%	224	79.29%	8	70.00%	20	99.92%		

nicely. However, the on-access component was a little less straightforward – it needed some compilation and access to the *dazuko* sources, which caused a headache and required calls to the developers for advice as the various requisites were set up. The *libdazuko* library, not built by default by the standard setup process, was also needed, but when at last everything was in place testing proceeded without further incident. Configuration, both of the command-line scanner and the on-access components, operated in a straightforward and standard fashion, with ample documentation available to guide the novice user.

Scanning speeds were about what I should have expected, my hopes of seeing testing times cut drastically as a result of using a pared-down operating system having quickly been dashed. On-access scanning speeds in particular were somewhat slower than I had hoped, doubtless due in large part to the test being run across the network. *Avast!*'s detection rates were little changed from previous tests, although detection for the single file-infector on the WildList which was missed last time around was added and the core set was covered without problems.

In the clean sets, however, a couple of items were mislabelled as malware, including one which has tripped up a series of products in the past year, and thus *Alwil* will

have to wait a little longer before reclaiming its place on the VB100 podium.

AVG Anti-Virus 7.5.51

ItW	100.00%	Polymorphic	73.89%
ItW (o/a)	100.00%	Linux	88.33%
Worms & bots	99.94%	Legacy	95.83%
File infectors	98.43%	False positives	0

AVG's product was considerably simpler to install, coming as a single .deb installer package which set everything up in a few moments, the majority of which were spent entering a licence key. Again using the *dazuko* file-hooking system, this time all the installer required was the kernel module to be in place. The design conformed to *Linux* norms, with straightforward syntax to the command-line scanner and the configuration files for the on-access monitor. Guidance and information was also ample and properly implemented.

Speeds were a little disappointing, even more so with scanning of all files and archives enabled, but detection



On-demand throughput	Archive Files				Binaries and System Files				Linux Files				Media and Documents				Other File Types			
	Default Settings		All Files		Default Settings		All Files		Default Settings		All Files		Default Settings		All Files		Default Settings		All Files	
	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)	Time (s)	Through-put (MB/s)
Alwil avast!	980	3.77	980	3.77	554	6.77	554	6.77	856	2.16	856	2.16	158	11.37	158	11.37	143	6.49	143	6.49
AVG Anti-Virus	2822	1.31	2910	1.27	832	4.51	1518	2.47	3638	0.51	5846	0.32	399	4.49	410	4.37	439	2.11	560	1.65
Avira AntiVir	57	64.31	751	4.92	276	13.58	290	12.91	2078	0.89	2362	0.78	159	11.28	160	11.19	165	5.62	166	5.57
BitDefender Security	1599	2.31	1599	2.31	612	6.12	612	6.12	1240	1.49	1240	1.49	163	11.00	163	11.00	183	5.06	183	5.06
Doctor Web Dr.Web	3739	0.99	3739	0.99	852	4.40	852	4.40	1576	1.17	1576	1.17	196	9.12	196	9.12	223	4.16	223	4.16
ESET Security	1110	3.33	1110	3.33	850	4.41	850	4.41	772	2.39	772	2.39	111	16.15	111	16.15	123	7.53	123	7.53
Frisk F-PROT	539	6.86	539	6.86	847	4.42	847	4.42	627	2.94	627	2.94	106	16.94	106	16.94	110	8.38	110	8.38
F-Secure Linux Security	4307	0.86	4307	0.86	975	3.84	975	3.84	2524	0.73	2524	0.73	323	5.54	323	5.54	345	2.69	345	2.69
Kaspersky Anti-Virus	3246	1.14	3246	1.14	685	5.48	685	5.48	1571	1.17	1571	1.17	188	9.52	188	9.52	207	4.46	207	4.46
MicroWorld eScan	4036	0.92	4036	0.92	615	6.09	615	6.09	2153	0.86	2153	0.86	256	6.99	256	6.99	275	3.36	275	3.36
Norman Virus Control	1228	3.01	1228	3.01	3335	1.12	3335	1.12	1697	1.09	1697	1.09	154	11.60	154	11.60	301	3.07	301	3.07
Quick Heal	957	3.86	957	3.86	187	20.04	187	20.04	1224	1.51	1224	1.51	135	13.29	135	13.29	106	8.69	106	8.69
Sophos Anti-Virus	61	60.13	2004	1.84	523	7.17	560	6.69	547	3.38	1476	1.25	99	18.14	194	9.24	63	14.62	249	3.71
Symantec AntiVirus	354	10.44	NA	NA	413	9.08	NA	NA	1351	1.37	NA	NA	198	9.05	NA	NA	211	4.38	NA	NA
VirusBuster SambaShield	345	10.72	346	10.69	524	7.15	524	7.15	621	2.97	621	2.97	102	17.63	102	17.63	104	8.91	104	8.91

rates were good, with no WildList samples missed and no false positives raised, and thus AVG earns a VB100 award.

Avira AntiVir for Linux 2.1.12-31

ItW	100.00%	Polymorphic	100.00%
ItW (o/a)	100.00%	Linux	66.67%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	0

Avira developed the *dazuko* system and continues to fund its maintenance and development. It is not surprising, therefore, that the company's product is among those making use of the file-hooking software.

The installation setup came as an archive file containing an install script, as well as the pre-built *dazuko* module – I suspect this addition is not generally provided for customers, but many *Linux* distributions come with the binary package available. The installer offered the delights of centralized management systems and graphical interfaces, which I was forced to turn down, and again the design, settings and documentation were excellent.

Scanning speeds this time were a little more impressive, and once again detection rates were superb, with most of the missed items merely being the result of rare file types



not being scanned with the default settings. With flawless coverage of the WildList and not a hint of a false positive, *Avira* easily qualifies for a VB100 award.

BitDefender Security for Linux 3.0.0.80505

ItW	100.00%	Polymorphic	100.00%
ItW (o/a)	100.00%	Linux	93.33%
Worms & bots	100.00%	Legacy	99.93%
File infectors	98.95%	False positives	0

BitDefender's product was another using the *.deb* system, this time with a built-in installation system too. This caused some issues initially as older versions of C++ libraries were required, which in turn required the installation of several other dependencies. Presumably on a fully networked system this would all have been handled by the package manager, reaching out to the web for any requirements.

Once over these hurdles things went very easily however, with a *Samba* VFS module used for the on-access component – this was the standard alternative to the *dazuko* system in the last *Linux* test and its operation proved simple, with a small change to the *Samba* configuration to point it at the new scanning object the only requirement.



In the previous *Linux* test several products using the VFS method encountered difficulties with speed and stability, but there were no such issues here, with things running along at excellent speeds without so much as a wobble until on-demand scanning of the archive set brought up a few segmentation-fault crashes. A handful of items were removed and the test was completed successfully, and the issue could not be reproduced in isolation. Detection was excellent, and without any samples missed in the WildList set and avoiding false positives, *BitDefender* also wins another VB100 award.

Doctor Web Dr. Web for Linux 4.44.0

ItW	97.55%	Polymorphic	100.00%
ItW (o/a)	97.55%	Linux	100.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	3

The *Dr.Web* product was a little more pared-down than the others, with a few simple .tgz archives which just needed extracting into the system root to drop their files into the right spots. After some teething problems with permissions – the result of inadequate perusal of the documentation on my part – things got trotting along nicely. I found the syntax of the command-line scanner a little quirky, but soon mastered it, along with the implementation of the on-access scanner SpIDerGuard, which again made use of *Samba*'s built-in VFS objects system.

The product's extreme thoroughness in analysing archives meant that scan times on some sets were rather long, but hugely detailed logs were produced, packed with information on the files which had been scanned. These included alerts on a range of 'riskware' and 'hacktool' products which I may not have wanted to have around had I been a genuine network administrator.

On more normal files speeds were very impressive, and detection rates were also extremely high, but once again a handful of items from the WildList set were not covered, and a couple of items in the clean set were mislabelled as malware, thus denying *Dr.Web* a VB100 award this time.

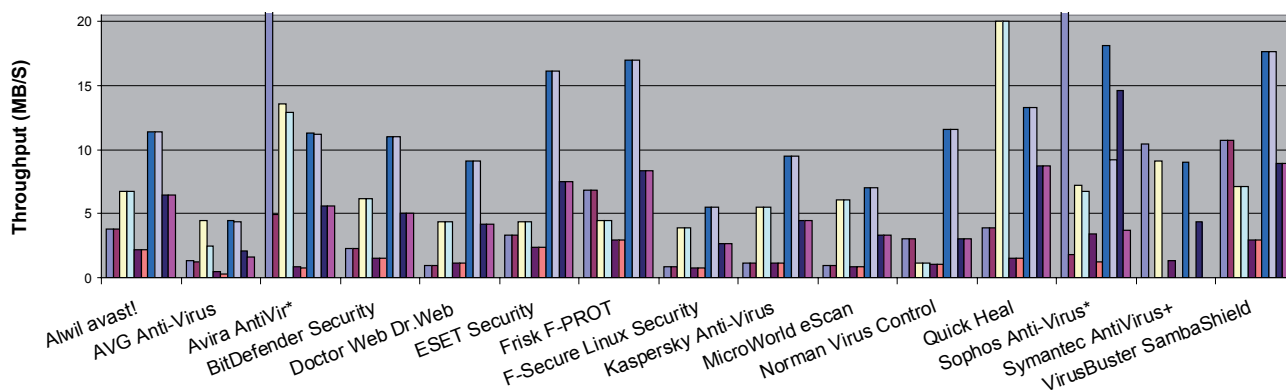
ESET Security 3.0.3

ItW	100.00%	Polymorphic	100.00%
ItW (o/a)	100.00%	Linux	100.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	0

The *ESET* installation process returned to the .deb package method, and proved fast and efficient. On-access scanning could be implemented using either *dazuko* or the *Samba* VFS path, and the latter was adopted at the request of the developers. This proved simple to get working once I had navigated my way



On-demand throughput



* Default archive scanning rate exceeds chart area
+ Full configuration not accessed

Archive files - default settings	Archive files - all files	Binaries and system files - default settings
Binaries and system files - all files	Linux files - default settings	Linux files - all files
Media and documents - default settings	Media and documents - all files	Other file types - default settings
Other file types - all files		

File access lag time	Archive Files				Binaries and System Files				Linux Files				Media and Documents				Other File Types			
	Default Settings		All Files		Default Settings		All Files		Default Settings		All Files		Default Settings		All Files		Default Settings		All Files	
	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)	Time (s)	Lag (s/MB)
Alwil avast!	972	0.26	973	0.26	554	0.13	555	0.13	856	0.19	857	0.19	158	0.06	159	0.06	143	0.10	144	0.10
AVG Anti-Virus	76	0.02	1791	0.48	534	0.13	814	0.20	2848	1.27	3465	1.60	358	0.17	398	0.19	362	0.34	476	0.46
Avira AntiVir	68	0.02	835	0.22	354	0.08	491	0.12	2506	1.08	2598	1.13	330	0.16	339	0.16	356	0.33	356	0.33
BitDefender Security	1700	0.46	1700	0.46	780	0.20	780	0.20	2865	1.27	2865	1.27	364	0.17	364	0.17	389	0.37	389	0.37
Doctor Web Dr.Web	2726	0.74	2726	0.74	980	0.25	980	0.25	2507	1.08	2507	1.08	342	0.16	342	0.16	359	0.33	359	0.33
ESET Security	2367	0.64	2367	0.64	1618	0.42	1618	0.42	1611	0.60	1611	0.60	208	0.09	208	0.09	152	0.11	152	0.11
Frisk F-PROT	386	0.10	386	0.10	677	0.17	678	0.17	1054	0.29	1055	0.29	150	0.06	151	0.06	150	0.11	151	0.11
F-Secure Linux Security	137	0.03	4342	1.17	735	0.18	1030	0.26	2337	0.99	2966	1.33	294	0.14	377	0.18	329	0.30	401	0.38
Kaspersky Anti-Virus	2564	0.69	2564	0.69	753	0.19	753	0.19	2070	0.84	2070	0.84	239	0.10	239	0.10	260	0.23	260	0.23
MicroWorld eScan	3376	0.91	3376	0.91	1141	0.29	1141	0.29	4477	2.15	4477	2.15	484	0.24	484	0.24	471	0.45	471	0.45
Norman Virus Control	102	0.03	NA	NA	652	0.16	NA	NA	1850	0.72	NA	NA	190	0.08	NA	NA	250	0.22	NA	NA
Quick Heal	35	0.01	NA	NA	240	0.05	NA	NA	1686	0.64	NA	NA	200	0.08	NA	NA	187	0.15	NA	NA
Sophos Anti-Virus	121	0.03	1173	0.32	581	0.14	645	0.16	1573	0.57	1699	0.64	227	0.10	230	0.10	251	0.22	257	0.22
Symantec AntiVirus	353	0.09	NA	NA	465	0.11	NA	NA	1713	0.65	NA	NA	221	0.10	NA	NA	243	0.21	NA	NA
VirusBuster SambaShield	73	0.02	NA	NA	591	0.14	NA	NA	1692	0.64	NA	NA	224	0.10	NA	NA	225	0.19	NA	NA

around the setup, and again the command-line scanner was a joy to operate.

Scanning speeds were not as eye-watering as usual, but they seemed much quicker on the infected sets, suggesting that the clean items were being subjected to some thorough probing. With excellent detection and no false positive issues, *ESET* storms its way to a record 50th VB100 award.

Frisk F-PROT Antivirus 6.2.1.4252

ItW 100.00% **Polymorphic** 100.00%
ItW (o/a) 100.00% **Linux** 100.00%
Worms & bots 100.00% **Legacy** 100.00%
File infectors 100.00% **False positives** 1

Installation of *F-PROT* took the simple method of extracting an archive onto the system and poking around inside it for the required tools and daemons. Man pages and other hints were plentiful and clear, and setup was a painless process, as was testing itself.

Having become accustomed to the dragged-out nature of the tests so far, *F-PROT*'s scanning speeds seemed lightning-quick, with detection rates equally remarkable. But, just as everything was looking rosy for *F-PROT*, a single item in the clean set – a rather specialist text editing tool – was labelled as a backdoor program, and *F-PROT* therefore fails to make the VB100 grade by a whisker.

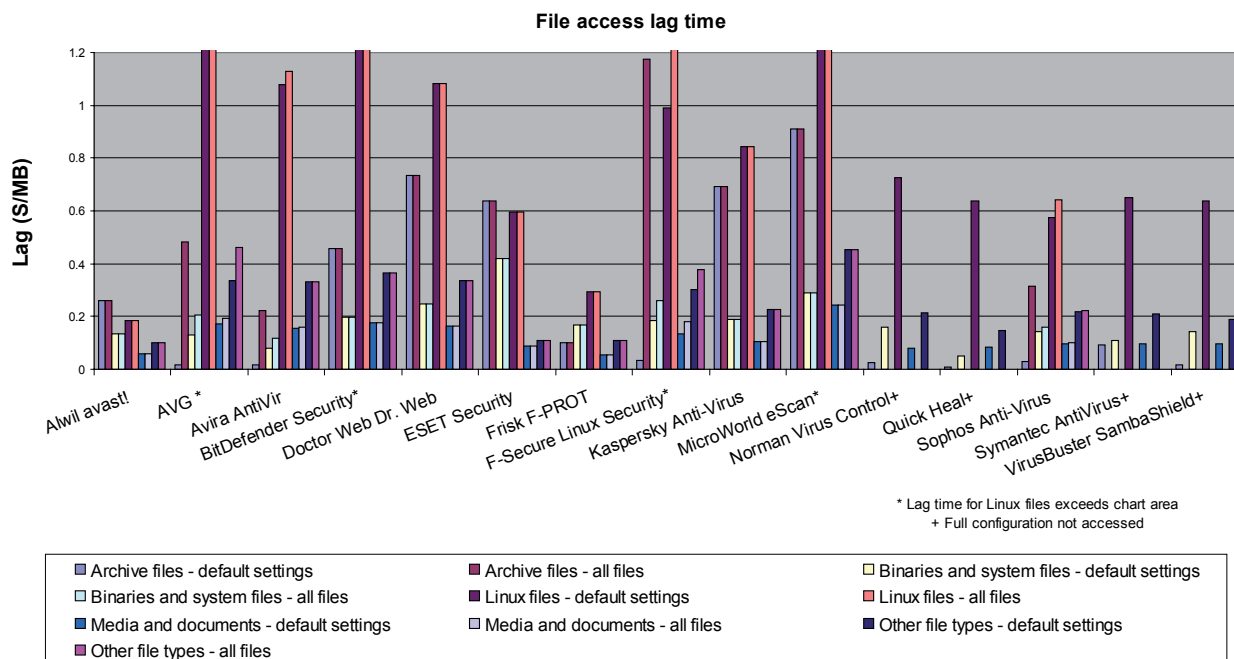
F-Secure Linux Security 7.00.71615

ItW 100.00% **Polymorphic** 99.88%
ItW (o/a) 100.00% **Linux** 100.00%
Worms & bots 100.00% **Legacy** 100.00%
File infectors 100.00% **False positives** 1

F-Secure's product is a lot bigger and shinier, with a complex installation process involving first setting up several dependencies, running the initial installer then running a secondary configuration program. Part of the reason for this bulkiness is the complexity of the product – in addition to the simple anti-virus scanner provided by most submissions this month, *F-Secure*'s product includes its own firewall and a series of intrusion-prevention and integrity-checking tools.

Getting things running was initially a little tricky, requiring in-depth perusal of a lengthy PDF manual included inside the install packages – of course, with no desktop environment on the test systems, this required copying it back to the client machine (and installing PDF viewing software) to read it. On first attempt the product claimed its on-access component was active, but it seemed to be having no effect, and the web interface – which appeared to be the only means of accessing much of the configuration – was inaccessible beyond the login page.

On second attempt things went much better, however – the on-access scanner, apparently based on a custom version of the *dazuko* technology, worked fine and the



web interface presented a pleasant and well-laid-out experience.

I operated on-demand scanning via the command line as usual, and other tests also proceeded normally after a few tweaks to the settings via the GUI. Scanning speeds were fairly languid once again, but scanning levels were thorough and detection equally in-depth. Just when all was looking up, the same tool that tripped up *F-PROT* was alerted on, this time being labelled an Ircbot trojan. This meant that *F-Secure* was also denied a VB100 award this month, and as the alert was marked as originating from the *AVP* engine, more upsets were expected.

Kaspersky Anti-Virus for Samba Servers for Linux 5.5

ItW	100.00%	Polymorphic	99.88%
ItW (o/a)	100.00%	Linux	100.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	1

Kaspersky uses the .deb package method for its install, but this time it seemed to do little more than place the required software in the right spots; exactly where these spots might be was left somewhat unclear. After some rummaging around I found the manual pages and linked them in with the man system, which shed some light on how to proceed. Post-install scripts had doctored my *Samba* configuration

to include the VFS on-access scanner, which was fairly simple to configure. The command-line scanner operated in a fairly normal way too, although it had an unwieldy title and required to be run as root to access its own configuration files.

Once things were up and running everything went fairly smoothly, with speeds not too bad in a slow month like this. Detection rates were excellent as ever, including complete coverage of the WildList, but as expected that pesky false positive cropped up once more and *Kaspersky Lab* fails to add another VB100 award to its collection.

MicroWorld eScan for Linux Server 2,0-16

ItW	100.00%	Polymorphic	99.88%
ItW (o/a)	100.00%	Linux	100.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	1

After the performance of the last few products, things did not bode well for *MicroWorld*, which is another product based on *Kaspersky's AVP* engine.

The installation process was another complicated monster, with an enormous list of dependencies thoughtfully provided by the developers. While much of the list could be acquired from repositories on the platform's install CD, many more items had to be scavenged from the Internet. Many of them seemed to relate to the graphics display,

Archive scanning depth		ACE	CAB	JAR	LZH	RAR	TGZ	ZIP	ZIP-SFX	EXT*
Alwil avast!	OD	X	X	√	X	X	√	√	√	√
	OA	X	√	√	√	√	√	√	√	√
AVG Anti-Virus	OD	X	√	√	X	√	X	√	√	√
	OA	X/2	X/√	X/√	X/√	X/√	X/√	X/√	X/√	√
Avira AntiVir	OD	X/√	X/√	X/√	X/√	X/√	X/√	X/√	X/√	√
	OA	X/√	X/√	X/√	X/√	X/√	X/√	X/√	X/√	√
BitDefender Security	OD	√	√	√	√	√	8	√	8	√
	OA	√	√	√	√	√	8	√	8	√
Doctor Web Dr.Web	OD	X	√	√	√	√	√	√	√	√
	OA	X	7	7	7	7	3	7	3	√
ESET Security	OD	√	√	√	√	√	5	√	√	√
	OA	√	√	√	√	√	5	√	√	√
Frisk F-PROT	OD	1	5	5	√	5	2	5	5	√
	OA	1	5	5	X	5	2	5	5	√
F-Secure Linux Security	OD	√	6	6	6	6	3	6	6	√
	OA	X/√	X/6	X/6	X/6	X/6	X/3	X/6	X/6	√
Kaspersky Anti-Virus	OD	√	√	√	√	√	√	√	√	√
	OA	√	√	√	√	√	√	√	√	√
MWTI eScan	OD	√	√	√	√	√	√	√	√	√
	OA	√	√	√	√	√	√	√	√	√
Norman Virus Control*	OD	X	X	√	√	X	√	√	X	√
	OA	X	X	X	X	X	X	X	X	√
Quick Heal*	OD	2	√	√	X	√	1	√	X	√
	OA	2	X	X	X	X	X	X	X	√
Sophos Anti-Virus	OD	X	X/5	X/5	X/5	X/5	X/5	X/5	X/5	√
	OA	X	X/5	X/5	X/5	X/5	X/5	X/5	X/5	X/√
Symantec Anti-Virus*	OD	X	X	3	3	3	3	3	3	√
	OA	X	X	3	3	3	3	3	3	√
VirusBuster*	OD	2	√	X	X	√	√	√	X	√
	OA	X	X	X	X	X	X	X	X	√

Key:

X - Archive not scanned

√ - Archives scanned to depth of 10 or more levels

[1-9] - Archives scanned to limited depth

X/√ - Default settings/thorough settings

*Increased archive handling options not accessed in some products

so I assumed they were required to support some kind of interface, which would not be required. Aided by detailed instructions provided by the developers I gathered them up and eventually managed to get all the *eScan* components, most of which were provided as .deb packages, to install and run.

Once everything was happy, and after another tweak to a *Samba* configuration file to activate a VFS object, the test chugged along at a fairly laid-back pace, the default settings being extremely thorough. In the end things went much as predicted, with excellent detection rates throughout and just that single pesky little file mislabelled in the clean set spoiling *MicroWorld's* hopes for a VB100 award.

Norman Virus Control 5.701

ItW	100.00%	Polymorphic	73.47%
ItW (o/a)	100.00%	Linux	100.00%
Worms & bots	100.00%	Legacy	99.00%
File infectors	99.15%	False positives	0

With *Norman* we returned once more to the simple and trusty method of dropping an archive load of files into the

filesystem root. This did a splendid job, setting everything up just so, including the man pages, which enabled fast and easy navigation of the configuration process. With the *dazuko* module loaded once again, everything looked set to go in record time.

Getting through the on-demand tests proved a breeze, thanks to the lucid instructions and logical controls. The on-access monitoring seemed solid and functional, although slightly lower in detection of some file-infecting viruses thanks to the *Sandbox* technology playing less of a role by default. Looking to change these settings, all the advice I could find referred me to a Java-based interface, but getting access to this would require considerable effort and time – which was running short.

Skipping the added speed measurements with full scanning enabled, a quick look through the logging showed that *Norman* had brought a run of recent misses to an end with excellent WildList detection and no false positives, bringing it proudly back to VB100 certified status.



Quick Heal 9.50

ItW	100.00%	Polymorphic	83.86%
ItW (o/a)	100.00%	Linux	66.67%
Worms & bots	99.87%	Legacy	93.95%
File infectors	98.43%	False positives	2

Quick Heal is another *dazuko* product, which also had a few other dependencies to fill. This proved to be a fairly straightforward job thanks to some tips from the developers.

The setup process was clear, helpful and even colourful, making a surprising and impressive difference to the clarity of an installation, which can often get swamped in a mass of samey text. Getting to grips with the command-line scanner proved a little less smooth, with a rather unusual syntax required including putting the path to files to be scanned before all other arguments. A GUI is also available, for those hedonists running glitzy desktop environments, but the command line served ably once its intricacies had been mastered.

The product lived up to its name with its scanning speeds, which were helped in the on-access test by not scanning archives by default. Upon investigating this, I found various options for the configuration of on-access logs and other sundries, but little concerning the actual types of files scanned. Of course, I may have been looking in the wrong place. Moving swiftly on to the results, I found detection rates at their usual decent level with excellent scores on the WildList and other more recent samples, but once more a couple of items in the clean set were alerted on and *Quick Heal* misses out on a VB100 award this month.

Sophos Anti-Virus for Linux 6.3.3

ItW	100.00%	Polymorphic	100.00%
ItW (o/a)	100.00%	Linux	65.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	0

Sophos's product is unusual in this test in using its own file-hooking setup. This goes by the name of *Talpa* and apparently, like *dazuko*, has been made open-source but is not as widely implemented. This gave rise to a few worries, as the platform under test is rather new and as yet not officially supported. However, after some confusion over which version I should be using, things went like a dream.



The product is supplied as a simple .tgz archive and an installation script which prepares and sets up everything very neatly, including pleasantly accessible documentation.

The command-line scanner uses pretty straightforward and humanly readable syntax, and scanning times were excellent with the bare, no-options settings. Tweaking them up a bit still produced good speed, and the on-access scanner was similarly zippy, although working out the rather less straightforward configuration system took a few moments. In the end, detection rates were top-notch, false positives absent, and *Sophos* put some upsets in recent tests behind it by winning a VB100 award with ease.

Symantec AntiVirus 1.0.4.516

ItW	100.00%	Polymorphic	100.00%
ItW (o/a)	100.00%	Linux	100.00%
Worms & bots	100.00%	Legacy	100.00%
File infectors	100.00%	False positives	0

I approached *Symantec*'s product with some dread, remembering a rather traumatic experience with its Byzantine configuration system in last year's *Linux* test. Searching inside the pair of archives provided I found a selection of .deb packages and some documentation in PDF files, which were shipped over to the workstation for some in-depth browsing.

Eventually, after a few false starts thanks to conflicting instructions at different points, things were up and running pretty solidly.

Next came the equally arduous task of navigating my way around the product, which was carried out mostly by means of passing arguments to the 'sav' command, which would then silently be followed by the scanning and monitoring daemons. This made monitoring the progress of scans rather tricky, having to rely mainly on watching the tail of the syslog – the only logging method available as far as I could tell – and keeping an eye on the hard drive activity lights.

On-access monitoring defaults to removing or disinfecting files, so to speed things along I delved bravely into the full glory of the configuration system, digging up secrets gleaned from tech support gurus for the last test. A listing of the configuration settings, which take the form of a registry-style database of keys, provided a little illumination, but its true meanings were far from clear. Passing in commands proved a lengthy and difficult process. Building up huge commands to pass in simple changes and



an absence of feedback to confirm an instruction had been accepted, required repeated trawls through the data to check changes had indeed occurred.

Once everything was under control, things moved along nicely, with excellent scanning speeds and the usual impeccable detection. Archive settings were a little low to measure fairly against others on the speed graphs, but changing them would have required more visits to the config system, and the resultant wear and tear on keyboards would have eaten heavily into my hardware budget. Leaving things as they were, *Symantec's* perfect detection scores across all test sets and absence of false positives earns it yet another VB100 award.

VirusBuster SambaShield 1.2.0_10

ItW	100.0%	Polymorphic	79.29%
ItW (o/a)	100.00%	Linux	83.33%
Worms & bots	99.91%	Legacy	99.92%
File infectors	99.21%	False positives	0

The final product on the list came from *VirusBuster* and proved a much more pleasant experience. Provided as a pair of archives with perl installation scripts, the setup ran through speedily and without difficulty. Another *Samba* VFS object managed the on-access side of things, and the layout and syntax seemed generally well thought out and sensible.



The tests zoomed through at an impressive rate, and detection levels showed continued improvement. Again a lack of time prevented in-depth investigation of the on-access configuration system to enable full archive scanning, but the only other problem encountered was the layout of the logging, keeping the scanned path on a separate line from detection information, which entailed a few extra stages to my results parsing. Beyond this minor quibble, though, the WildList was covered without problems, and the rest of the sets handled pretty well too, and without any false positives either *VirusBuster* earns a VB100 award.

CONCLUSIONS

Once more the *Linux* test has proved to be the domain of the hardened VB100 experts, the small list of products participating in this test consisting entirely of names made familiar from consistent and dogged appearances in the test month after month. A few regulars were notable by their absence, with some of the larger, more corporate-focused

companies yet to implement support for the platform selected.

After a string of low-scoring tests I had hoped that this month might finally see a clean sweep with all products passing, and as far as the WildList went we nearly made it, with only one product having trouble in this area. Once again, however, the test's strict false positive rules played a major part, with just four files scuppering the chances of a VB100 award for six of the products.

Beyond the basics of the scores, the products themselves displayed a dizzying variation in style and implementation, with some remaining extremely simple while others have expanded their functionality in a range of new ways. Both paths proved capable of providing stable, rapid and usable products as well as confusing, sluggish and wobbly protection, with documentation – or at least accessing it – being the most significant factor as far as ease of use was concerned. Of course, submissions were not necessarily made in the same format as paying customers would receive, and the likelihood of more obvious installation instructions and user manuals would make a big difference in some cases. With a little work, however, all products were made to function sufficiently well to get through the tests, and all provided a decent level of configurability, albeit in some cases in a rather bizarre and arcane fashion.

The added complexity of the installation and navigation of various products meant that this month's comparative was not the quick and restful experience I had hoped for between two much larger tests. It has highlighted the pace with which most products are keeping up with our test sets, and the need for more rapid expansion of those test collections to provide a more accurate gauge of their capabilities. Hopefully, June will grant time to ensure the test sets are well enlarged in time for the forthcoming *XP* comparative, due to commence at the start of July and to appear in the August issue of *VB*. Perhaps that test, which I expect to break the 40 product mark, will finally see that clean sweep with no failures – I can but hope.

Technical details:

Tests were run on identical machines with *AMD Athlon64 3800+* dual core processors, 1 GB RAM, 40 GB and 200 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy drive, running *Ubuntu Linux 8.04LTS Server* edition.

Client machines had 1.6GHz *Intel Pentium* processors with 512 MB RAM, 20 GB dual hard disks, CD-ROM and 3.5-inch floppy drive running *Microsoft Windows XP Professional SP2*, connected via *Samba 3.0.28a*.

END NOTES & NEWS

Hacker Halted USA 2008 takes place 1–4 June 2008 in Myrtle Beach, SC, USA. The conference aims to raise international awareness of the need for increased education and ethics in information security. Hacker Halted USA delegates qualify for free admission to the Techno Security Conference which runs concurrently. For more details see <http://www.hackerhalted.com/>.

The 20th annual FIRST conference will be held 22–27 June 2008 in Vancouver, Canada. The five-day event comprises two days of tutorials and three days of technical sessions where a range of topics of relevance to teams in the global response community will be discussed. For more details see <http://www.first.org/conference/>.

SANS InfoSec for Business Executives will take place 1–2 July 2008 in Seattle, WA, USA. The 1.5-day course is designed for executives who need to understand what information security is, and to provide a better understanding of information security needs. For more information see <http://www.sans.org/seattle08/>.

The SecureAmsterdam conference on emerging threats takes place 15 July 2008 in Amsterdam, the Netherlands. For details see <https://www.isc2.org/cgi-bin/events/information.cgi?event=66>.

SANSFIRE 2008 takes place 22–31 July 2008 in Washington, DC, USA. The course schedule for SANSFIRE 2008 features a full line-up in the disciplines of audit, security, management and legal as well as new courses with a focus on penetration testing, malware analysis and removal, and secure coding. For more information see <http://www.sans.org/sansfire08/>.

The 17th USENIX Security Symposium will take place 28 July to 1 August 2008 in San Jose, CA, USA. A two-day training programme will be followed by a 2.5-day technical programme, which will include refereed papers, invited talks, posters, work-in-progress reports, panel discussions, and birds-of-a-feather sessions. For details see <http://www.usenix.org/events/sec08/cfp/>.

Black Hat USA 2008 takes place 2–7 August 2008 in Las Vegas, NV, USA. Featuring 40 hands-on training courses and 80 Briefings presentations. This year's Briefings tracks include many updated topics alongside the old favourites including zero-day attacks/defences, bots, application security, deep knowledge and turbo talks. Online registration is now open. For details see <http://www.blackhat.com/>.

VB2008 will take place 1–3 October 2008 in Ottawa, Canada. Presentations will cover subjects including: sample sharing, anti-malware testing, automated analysis, rootkits, spam and botnet tracking techniques, corporate policy, business risk and more. Register online at <http://www.virusbtn.com/conference/vb2008>.

Black Hat Japan 2008 takes place 7–10 October 2008 in Tokyo, Japan. For full details see <http://www.blackhat.com/>.

Net Focus UK 2008 takes place 8–9 October 2008 in Brighton, UK. The event deals with issues of security, personnel, compliance, data privacy, business risk, e-commerce risk and more. For details see <https://www.baptie.com/events/show.asp?e=160&xyzy=2>.

The third APWG eCrime Researchers Summit will be held 15–16 October 2008 in Atlanta, GA, USA. eCrime '08 will bring together academic researchers, security practitioners and law enforcement to discuss all aspects of electronic crime and ways to combat it. For more information see <http://www.antiphishing.org/ecrimeresearch/>.

The SecureLondon Workshop on Computer Forensics will be held 21 October 2008 in London, UK. For further information see <https://www.isc2.org/cgi-bin/events/information.cgi?event=58>.

RSA Europe 2008 will take place 27–29 October 2008 in London, UK. For full details see <http://www.rsaconference.com/2008/Europe/>.

CSI 2008 takes place 15–21 November 2008 in National Harbor, MD, USA. A call for papers is now open. Online registration will be available from June. See <http://www.csiannual.com/>.

AVAR 2008 will be held 10–12 December 2008 in New Delhi, India. A call for papers has been issued, with a submission deadline of 15 July. For more details see <http://www.aavar.org/avar2008/>.

ADVISORY BOARD

Pavel Baudis, Alwil Software, Czech Republic
Dr Sarah Gordon, Independent research scientist, USA
John Graham-Cumming, France
Shimon Gruper, Aladdin Knowledge Systems Ltd, Israel
Dmitry Gryaznov, McAfee, USA
Joe Hartmann, Microsoft, USA
Dr Jan Hruska, Sophos, UK
Jeannette Jarvis, Microsoft, USA
Jakub Kaminski, Microsoft, Australia
Eugene Kaspersky, Kaspersky Lab, Russia
Jimmy Kuo, Microsoft, USA
Anne Mitchell, Institute for Spam & Internet Public Policy, USA
Costin Raiu, Kaspersky Lab, Russia
Péter Ször, Symantec, USA
Roger Thompson, CA, USA
Joseph Wells, Lavasoft USA

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

Editorial enquiries, subscription enquiries, orders and payments:

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

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

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

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

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

VIRUS BULLETIN © 2008 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England. Tel: +44 (0)1235 555139. /2008/\$0.00+2.50. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

vb Spam supplement

CONTENTS

- S1 **NEWS & EVENTS**
- S1 **FEATURE**
Is your spam filter really adaptive?
(Probably not)

NEWS & EVENTS

ISRAEL OUTLAWS SPAM

In the month in which spam officially turned 30, Israel's legislature passed a new ruling that outlaws the sending of unsolicited messages by email, telephone, SMS or fax. The Knesset approved the bill – which will see spammers facing fines of up to NIS 200,000 (approximately £31,000) – late last month.

Israel's new anti-spam legislation is based on an opt-in approach, in which advertisers may not contact individuals unless they have agreed specifically to receive bulk mail/solicitations. Exceptions are made for those who make purchases from and send their details to a company (which may then send unsolicited messages to the individual unless they have specifically requested not to receive such communication) and for bulk mailing carried out for political or charity purposes.

Once the new law is enacted, individuals will be able to sue for up to NIS 1,000 (approximately £155) or file a complaint with the police, who will follow up with a criminal charge leading to stiffer penalties.

The new legislation is expected to take effect in six months.

EVENTS

The 13th general meeting of the Messaging Anti-Abuse Working Group (MAAWG) will be held in Heidelberg, Germany, 10–12 June 2008. The meeting is open to MAAWG members only. The 14th general meeting (also members only) will take place 22–24 September 2008 in Harbour Beach, FL, USA. See <http://www.maawg.org/>.

CEAS 2008 will take place 21–22 August 2008 in Mountain View, CA, USA. For more information about the event see <http://www.ceas.cc/2008/>.

FEATURE

IS YOUR SPAM FILTER REALLY ADAPTIVE? (PROBABLY NOT)

Jonathan Zdziarski
Secure Computing, USA

It has been several years since I first entered the spam scene, and during this time I have observed carefully how machine learning has degenerated – seemingly unnoticed – into a cat and mouse game between spammers and filter authors.

Ironically, this is what we thought we had avoided when statistical content filters went mainstream. At the time, most of us believed that Bayesian filters had the moxie to fight our battles for us. This was because these filters adapted to new forms of content on their own, leaving our customers only to push the 'spam' button. What used to mean having to update rules in heuristic filters became a service performed automagically by our statistical classifiers instead. This technique did, in fact, work for several years, and as a result filter authors became lazy. Reality has hit us in recent years, though, and forced many of us to sit up and observe the different spammer techniques that have effectively (and in a seemingly demoralizing manner) rendered many adaptive filters helpless. Sure, Nigerian diplomat spam isn't bothering us any more, but image spam, ASCII spam, spear phishing, and (in one case) Klingon spam seem to put our spam filters through hell.

BLINDING THE PARSER

As mentioned, adaptive filters are capable of learning new forms of content. This was the holy grail of geek research when these filters first hit the scene. The inherent problem has been how to define what exactly content is and how to present it to the classifier so that the filter can adapt to it.

One key component of every spam filter is its parser (or tokenizer). The parser is responsible for 'reading' incoming mail and turning it into a set of features that the filter uses to build a hypothesis space (its logic). In its simplest operation, the parser might simply pull out every word found in an email, making the word selection based on where the spaces are, and call each word a feature. The parser represents the eyes of a spam filter, and nearly every recent attack by a spammer can be traced back to a weakness in parsing.

Whether spammers are smart enough to target the parser intentionally is unknown, but the likely reason that so many flaming arrows are shot at it is because it is the weakest part of even a well-written spam filter. Spammers have long given up trying to attack other parts of the filter, having become wise to the fact that it is very difficult to attack mathematics. Throwing random words or excerpts from books into an email has historically proven a glorious waste of time, and so in recent years, spammers have turned their attention from *confusing* the spam filter to trying to *blind* it instead.

In order to blind a spam filter, you have to know how one sees. Unlike a classifier's machine-learning components, the spam filter's parser is one of the few pieces that is written to have a predetermined set of rules – parsing rules, that is. In other words, the filter's creator hasn't *trained* the spam filter how to read, but rather *told* it how to read. Since every copy of a particular piece of spam-filtering software will read in the same way, spammers can easily – through trial and error – figure out how to hamper the filter's ability to do its job. If you can prevent the parser from distinguishing content, you can prevent it from making any sense of it. In human terms, this would be the equivalent of speaking Pig Latin to a human who was just learning the English language. While thoroughly amusing, it would take the human some time before realizing they should smack you. A well-written parser attack can keep the computer in a confused haze for long enough to get spam through.

To add insult to injury, parsers also lack the practical ability to read more than a few different languages effectively. Most are limited to parsing the character sets for which they were designed, as many languages use basic rules such as white space to determine word breaks. This presents a rather difficult problem when trying to classify mail in Chinese or Korean, which has no white space, or when analysing languages that have special accents. Moreover, if you're classifying PDF files or other quasi-binary content, the filter will have no way of parsing its binary components intelligently. In order to account for more complex languages and formats, the filter author himself would need to have foreknowledge of what languages he is supporting. Consider the following parse-o-grams, and you can get an idea of what Chinese email must look like to an English-‘speaking’ spam filter:

<u>THEREDWELLSAMISSLATE</u>	<u>ENDANGERSPARSEAMANSWORDS</u>
THE RED WELL, SAM IS LATE	ENDANGER! SPAR, SEAMAN SWORDS!
THERE DWELLS A MISS, LATE	END ANGERS; PARSE A MAN'S WORDS
(G. Sinnamon)	(D. Higgs)

Even though machines have proven their amazing capacity for learning, we still seem to stick with using static parsing

rules in an otherwise statistical classifier. However, the computer can teach itself how to read better than the human who programmed it can, if only given the chance.

ADAPTIVE PARSING

Adaptive parsing is a technique applied to a classifier's parser to allow it to learn the most effective way to parse a corpus of mail. Notice I didn't say the best way or the most intelligible way, but the most effective. 'Effective' here could be defined as 'rendering the most descriptive data' to the rest of the classifier. This is done by instructing the parser to measure the quality of its own output and reprogram itself when necessary. Data can be considered to be high quality when its occurrence within the corpus results in it having a very high or a very low probability – something that could potentially affect the result of the classification. On a Bayesian filter, good quality data could be considered to be somewhere in the neighbourhood of 0.0–0.1 and 0.9–1.0.

To find the best parsing technique, a hypothesis space of parsing semantics is built. From this, the parser can measure which techniques work best for whatever data has historically been presented. If the filter finds that it is not generating as useful data as it could, then it can reprogram itself dynamically to use a different set of parsing techniques. The change of techniques will cause the data to be extracted differently – inevitably either improving or degrading its results.

The most simple and natural form of parsing technique is simply distinguishing between a token separator and a constituent character. As a simple example, let's take all possible byte values (that's 0x00 to 0xFF) and turn them into a hypothesis space. These represent the 256 possible bytes that could be used by the filter as either token separators or constituent characters. Most spam filters will generally use only a handful of these, such as white space, punctuation, and the like. Here, all 256 possible characters are analysed within our hypothesis space, so we can later choose the ones that we think will give us the best results. In reality, this could be extended to include wide characters, or if you want to get fancy, kGrams of bytes.

As mail comes in, each feature is analysed by whatever the live parser configuration is using (we'll get to bootstrapping shortly). As part of the filter's basic operation, it will assign each feature a probability. A Bayesian filter might end up leaving you with:

THE	0.55
HUGE	0.80
FICKLE	0.02
FOX	0.03
LEFT	0.97

Let's make some observations about this data. Using our example thresholds of 0.0–0.1 and 0.9–1.0, the letter 'F' seems to appear only in interesting data – that is, the words 'FICKLE', 'FOX', and 'LEFT'. Similarly, the letter 'H' only appears in uninteresting data: 'THE' and 'HUGE', which both have less than impressive probabilities. The letter 'E' appears equally in both, as does the space character, which is unseen.

If we were to use only this data to write a set of parsing rules, we could easily put together an order of preference from uninteresting characters to interesting ones:

Uninteresting → → → Interesting
 H U [SP] E T O X F

To program a parser, you would naturally want to use the characters that were found to be in the least interesting data as token separators first. Why? Because characters like 'F' and 'X' have a very high probability of appearing in useful data – you want those characters to be constituent characters.

The parser can now be reprogrammed to use the space, 'H', 'U', and 'E' for the next message processed. This will cause the parser to take otherwise uninteresting data and 'read' it differently in a way that might render more useful tokens. If successful, the characters present in these new tokens will rise to the top as more interesting, and be used as constituent characters. If the data it generates becomes less interesting, then an even less useful set of characters will be used as token separators.

Of course, the data you wind up with may make less sense to a human, but what's important is that the classifier sees it to be of great interest. Here are a few real-world examples:

```
[0.940828] igh (105s, 2i)
    l-igh, High, H-IGH Interest Mortgage.
[0.990000] $888 (15s, 0i)
    Yup, dollar signs are more useful.
[0.990000] ional_Inc.+Now (6s, 0i)
[0.990000] s0r+C|ubs (12s, 0i)
```

Variants of spam that would otherwise be 'uninteresting'.

In the statistics world, we're essentially assigning a probability to each character for each token the parser examines. The probability can be assigned to each character using Paul Graham's original idea in 'A Plan for Spam' (<http://www.paulgraham.com/spam.html>):

$$P_{\text{DELIM}} = \frac{\text{LOW}_{\text{DELIM}} / \text{LOW}_{\text{TOTAL}}}{(\text{LOW}_{\text{DELIM}} / \text{LOW}_{\text{TOTAL}}) + (\text{HI}_{\text{DELIM}} / \text{HI}_{\text{TOTAL}})}$$

Instead of measuring the probability of spam vs. non-spam, we're now measuring the probability that the character will appear in useful data. And, unlike its original use where totals were based per message, the totals here are assessed on a per-token basis.

BOOTSTRAPPING

Every time a message is processed, features are extracted using whatever the least effective set of characters happen to be, and the rest are treated as constituent characters. These could be the best *N* candidates or a specific threshold could be used to determine how many characters to use. What we haven't figured out yet is how to get this thing started. There are a few schools of thought with respect to bootstrapping:

The first is to use a predetermined set of delimiters until enough data has been recorded to be confident in programming the parser adaptively. The drawback of this approach is that we don't know whether the default delimiter set will be able to parse effectively at all.

The second approach is simply to assign random probabilities, in much the same way as a neural network is initialized, and let the parser work out its own programming with training. Depending on the corpus, this could work very well for certain character sets.

Other techniques may work better. The idea is to get the parser going with some instruction on how to parse the first messages. This will lead the parser down the road of generating better programming for itself.

END RESULT

The philosophy here is to teach the parser how to parse by measuring its own ability to generate useful data. Statistically speaking, we are trending the presence of good data with relevant parsing rules. However, what goes into the hypothesis space could really be anything. Human-based rules for parsing certain languages could easily be included, as could other large sets of parsing semantics. With a little more complexity, this could be used to parse binaries statistically. Instead of measuring just what characters to parse on, one could also measure whether to parse before, on, or after the character, giving 768 possible parsing rules to calculate.

Eventually, the parser ends up with rules that look nothing like human rules, giving the filter data that looks nothing like human data – but if the classifier can make a more informed classification as a result of this, then that's perfectly all right with me.

This article is based on a paper presented at the 2008 MIT Spam Conference. See <http://spamconference.org/>.