SEPTEMBER 2008

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

E-ZINE DOES IT

Peter Ferrie begins a series of analyses of viruses contained in the long-delayed (and probably last of its kind) EOF-rRlf-DoomRiderz virus zine.
page 4

MISSING CODEC

Christoph Alme and his colleagues walk through the complete attack process of the Win32.ASF-Hijacker.A trojan, which has two different sets of victims: those affected by the trojan itself and those affected by media files altered by the trojan.
page 7

LEARNING BEHAVIOUR

Richard Ford and William Allen, both teachers of malware courses at Florida Institute of Technology, present their views on the teaching of virus writing as part of security courses.
page 11

vbSpam supplement

This month: anti-spam news and events, and Terry Zink presents the first in a two-part series of articles on one of the hot issues in the world of spam filtering: backscatter.

'Once bitten does not, it seems, make twice shy for spammers.'

Helen Martin, Virus Bulletin

DOES THE PUNISHMENT FIT THE CRIME?

In July a man who sent more than 50,000 spam emails an hour and who has been a known spammer since 1999 was sentenced to 47 months in jail after pleading guilty to charges of fraud, spamming and tax evasion. He was also fined a little over $700,000.

Prosecutors had been hoping for a stiffer sentence and had requested Robert Soloway be sent to prison for nine years, taking into account both the scale of his spamming operations and the fact that he had previously been investigated for spamming activities. This led me to wonder, what *is* a suitable punishment for a convicted spammer?

Jeremy Jaynes, the first spammer convicted in a felony case in the US (in 2004), was sentenced to nine years in prison after he was found guilty of sending several million messages in a two-month period. Jaynes is said to have made $750,000 a month at the height of his activities, with a net worth of up to $24 million.

Jaynes's case preceded the introduction of federal CAN-SPAM regulations, which allow for a judge to consider profits if financial damages are unclear, and in December 2007 Min Kim became the first convicted spammer to have his sentence lengthened due to high earnings from his spamming activities. The judge recommended that Kim should be imprisoned for 30 to 37 months rather than the 24 to 30 it would otherwise have been, because he had recorded profits of $250,000 from spamming.

While I abhor the activities of spammers and spend an infuriating amount of my time pressing the 'delete' button to rid my inbox of their issue – on occasion accidentally filtering out genuine messages due to a trigger happy (or fatigued) delete finger – I have to wonder whether they really pose such a significant danger to society that they should be locked up for years (while tax payers' money keeps them warm, fed and watered).

Of course the sentencing of spammers is rarely straightforward. In many cases the crimes they are charged with are wider ranging than 'just' the contravention of anti-spam laws. Spammers are often involved in such other nefarious activities as ID theft, distribution of malware, money laundering and so on, all of which quite rightly increase the severity of their sentencing. But where straightforward spamming is concerned incarceration does *almost* seem an unnecessary use of resources – for criminals who apparently pose little physical danger.

Many have suggested that spammers should simply be punished financially – as one *VB* reader said 'being locked in a financial prison is just as debilitating as jail but far less expensive to the tax payer'. There have been many lawsuits resulting in spammers being ordered to pay fines, but rarely to the extent that they render the spammer incapable of continuing their activities. In December 2003, NY Attorney General Eliot Spitzer said: 'We will drive [spammers] into bankruptcy, and therefore others will not come into the marketplace to take their place', yet five years on the marketplace is flooded with spammers.

The cases of Scott Richter and Sanford Wallace demonstrate that once bitten does not, it seems, make twice shy for spammers. Richter was sued by both *Microsoft* and the New York Attorney General in 2003, reaching a $40,000 settlement with the Attorney General and eventually agreeing to a $7 million settlement of the *Microsoft* case. Yet in 2007 Richter was back in court being sued by *MySpace* for more spamming activities (an arbitrator awarded *MySpace* $4.8 million in damages). Similarly, Wallace came to prominence as a prolific spammer in the mid 1990s, but in 1998 announced his retirement from the spamming business after facing lawsuits from *AOL* and *CompuServe*. Just last year *MySpace* sued Wallace for – you've guessed it – spamming and phishing activities.

Clearly financial penalties have not thus far proved a particularly strong deterrent, and I am left to conclude that there is indeed a need for spammers to face the prospect of a prison sentence – and for spammers such as Soloway, Jaynes and Kim to be held up as examples to send a firm message of discouragement to would-be spammers.

NEWS

ADVANCE DIARY DATES: VB2009

VB is pleased to announce that VB2009 will take place 23–25 September 2009 at the Crowne Plaza in Geneva, Switzerland. Reserve the dates and start making your travel plans now!

If you are interested in becoming a sponsor, or require any more information about VB2009, please contact us by emailing conference@virusbtn.com.

MCAFEE SUED FOR WEB WARNINGS

McAfee is being sued over its popular *SiteAdvisor* system, which rates web domains for security and privacy risks and is implemented in search results produced by *Yahoo!*. The security firm is being sued by a website its *SiteAdvisor* has labelled with the 'red-for-danger' mark.

The people behind *7search.com* claim their site is clean and have taken their grievances to a court in Illinois, demanding damages and enforced retraction of the alleged slur. However, wording on the *SiteAdvisor* page makes it clear that the label is based on user reports, and makes no direct accusations against the site. Numerous reporters have pointed out links between *7search* and previous, highly dubious ventures, including toolbars which claimed to speed up browsing but also gathered information on surfing habits. The current *7search* offering has been accused of hiding sponsored links amongst genuine search results.

Some commentators have argued that the case threatens the ability of security firms to properly protect their customers, while others think little of the suit's chances. Like most security products, *SiteAdvisor* has its share of false positive incidents, and *VB* has heard from several firms who have felt their own businesses threatened after red warning links from *SiteAdvisor* significantly reduced traffic to their websites.

One website owner, Julian Moss of *Tech-Pro.Net*, spent over a week watching his traffic dry up after a file linked to from his site (a product produced by *PC Tools*, recently acquired by *McAfee*'s arch-rival *Symantec*) produced a false alarm. He made little progress with the *SiteAdvisor* complaints procedure: 'Only a strong threat of legal action appeared to spur *McAfee* into action to investigate our complaint, confirm the file was harmless and remove our site from the blacklist,' he said. 'Even then, it was several more weeks before sites that received a red alert because they linked to ours were cleared.'

It will be interesting to see whether in this latest case, with what looks like a little more justification on its side, *McAfee* will stand up to the legal challenge.

Prevalence Table – July 2008

Malware	Type	%
NetSky	Worm	21.11%
Agent	Trojan	16.55%
Zbot	Trojan	10.96%
Bifrose/Pakes	Trojan	8.94%
Mytob	Worm	7.85%
Virut	Virus	5.47%
Suspect packers	Misc	3.60%
OnlineGames	Trojan	3.00%
Mydoom	Worm	2.95%
Cutwail/Pandex/Pushdo	Trojan	2.54%
Iframe	Exploit	2.49%
Bagle	Worm	2.20%
Mdropper	Trojan	1.91%
Lineage/Magania	Trojan	1.26%
Small	Trojan	1.07%
Grew	Worm	0.91%
Zlob/Tibs	Trojan	0.88%
Heuristic/generic	Trojan	0.82%
Delf	Trojan	0.81%
Zafi	Worm	0.76%
Sality	Virus	0.71%
Rays/Traxg	Worm	0.59%
Mywife/Nyxem	Worm	0.36%
Inject	Trojan	0.21%
Stration/Warezov	Worm	0.19%
Klez	Worm	0.17%
Bagz	Worm	0.15%
Alman	Worm	0.14%
Qhost	Trojan	0.13%
Buzus	Trojan	0.12%
Womble	Worm	0.10%
Heuristic/generic	Misc	0.09%
Parite	Worm	0.09%
Others[1]		0.87%
Total		100%

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

MALWARE ANALYSIS 1

PROPHET AND LOSS

Peter Ferrie
Microsoft, USA

The release of the long-delayed EOF-rRlf-DoomRiderz virus zine probably marks the last of its kind. While the quality is not terribly high, there are some viruses of interest. A series of analyses in alphabetical order begins with this one: W32/Divino.

I'M A LOCAL

The virus begins by storing the selector of the local descriptor table in the ImageBase field in the PEB, and then reading four bytes and checking if the result is non-zero. A non-zero result should always occur, because the top half of the ImageBase field will remain untouched and non-zero. This might be an anti-emulator trick for an emulator that stores four bytes instead of two. However, it seems more likely that what the virus author had in mind was to read only two bytes and detect whether the local descriptor table (LDT) is in use, but had to reverse the condition because of the extra bytes that the virus reads. The use of the LDT is a characteristic of virtual machines such as *VMware* and *VirtualPC*, along with *Norman*'s *SandBox*.

In any case, if the result is zero, the virus attempts to continue execution, but without decrypting itself first. When that happens, the virus crashes and the application terminates. If the result is non-zero, then the virus decrypts the first stage of its body and attempts to transfer control to it, using an address that was calculated from values in the PE header at the time of infection. This means that the virus is not aware of 'Address Space Layout Randomization' (ASLR). If the infected file was built to be ASLR-aware, then the virus will crash and the application will terminate.

UN-SafeSEH

The first stage of the virus registers a structured exception handler, then intentionally causes an exception. This is an old anti-debugging trick which any good debugger can skip easily enough. Since the handler appears immediately after the call to the anti-debugging routine, it's a simple matter to step over the call and continue execution. However, the virus is not aware of 'SafeSEH', which overrides the legacy structured exception handling. If the infected file was built with SafeSEH, then the exception that the virus raises will cause the application to exit, because the exception address will not match any known address.

The virus unregisters the handler, copies a decryptor to the stack, and then attempts to execute the decryptor from there. The virus is not aware of 'Data Execution Protection' (DEP). If the infected file was built to be DEP-aware, then the virus will crash and the application will terminate.

BYTE, BYTE BABY

The virus retrieves an address from the stack that points within the kernel32 BaseThreadInitThunk() function. Using this as a starting point, the virus performs a brute-force search in memory for the 'MZ' header. The search is performed byte by byte, rather than on 64KB boundaries, making it slow and inefficient. The virus does not register a structured exception handler for this operation. As a result, the technique fails on *Windows Vista64*. This is because the kernel32.dll in *Windows Vista64* uses a 64KB section alignment, so the region between the file header and the first section is not mapped. Any attempt to access this memory will cause an exception which is not intercepted by the virus. If an exception occurs, the virus will crash and the application will terminate.

ONWARD AND FORWARD

In the event that everything is okay, the virus calculates a pointer 4KB below the current stack pointer value. The virus author assumed that it would remain untouched but this is not always the case, as we will see below. The virus resolves a set of API addresses from kernel32.dll that are required to infect files. The resolver uses checksums instead of names. The list includes GetLastError(), but this is never used. In fact, almost one third of the APIs that are resolved are not used. The reason for not using GetLastError() is clear. It is because on *Windows XP*, the function is forwarded to ntdll.dll, so the address that is exported from kernel32.dll does not point directly to the function. The lack of forwarding support is a common problem for export resolvers in virus code, though the problem is not limited to viruses. There are many runtime packers that also use the checksum technique, which also do not support export forwarding. The reason for not using some of the other APIs is less clear, but the fact that some of them would be used to start a new process and watch its execution suggests that an alternative spreading mechanism was planned but not implemented.

The virus resolves a set of API addresses from ws2_32.dll, some of which will be used to perform a denial-of-service attack as part of the payload. Some of the APIs are not used, but could be used to form the basis for a remote control mechanism, or perhaps an auto-updating capability which may also have been planned but not implemented. The virus also resolves a set of API addresses from shell32.dll and

user32.dll, which will be used to perform some actions on the clipboard.

HANGING BY A THREAD

At this point, the virus copies back the bytes replaced by the first decryptor and creates a thread to run the host code. This allows the virus to achieve per-process residency, however this behaviour can be considered a bug. The problem is that if the host code terminates, the virus code will be forcibly terminated, too. This can result in an interrupted infection and a corrupted file.

The virus retrieves the first four bytes of the GetProcAddress() function and compares it to the 'enter 0,2' instruction. The virus wants to exit if there is a match. This might be an anti-emulator detection. However, since the 'enter' instruction is only three bytes long, the comparison will probably always fail. This behaviour appears to be a bug.

IT'S PAYBACK

The virus carries a payload whose trigger is the execution of an infected file on the 28th day of any month. The date is acquired by allocating a buffer requesting the date format, then comparing the contents to '28'. There is a bug in this routine, however, which is that the buffer is never freed.

The payload exists in two parts. The first part of the payload displays a message box. The message title is the two-digit date, so it is always '28'. The message body is:

```
Win32.Divinorum
Code By Fakedminded/EOF-Project
Mikko cut ur ponytail!
```

The 'Mikko' in the text is presumed to be a reference to *F-Secure*'s Mikko Hyppönen, who happens to wear his hair long.

The second part of the payload attempts to perform a denial-of-service attack on *F-Secure*'s European website. However, now the stack problem appears. The IsValidLocale() function, which is used by the MessageBox() function, uses so much of the stack on *Windows Vista* that it corrupts the API table. The result is that on *Windows Vista* the rest of the payload crashes, and the application terminates.

The virus author appears to be aware of the general problem, since the WSASocket() function also requires a lot of stack, but the virus saves and restores the stack state in order to survive that call.

The denial-of-service attack is effectively limitless, since it uses a 'loop' instruction which relies implicitly on the value in the ecx register. The ecx register is set as a side effect of the call to the Sleep() function. The value is the return address of the call to the EH_epilog() function, which is always greater than 1.

WINNERS AND LOSERS

The virus allocates some memory to hold the current directory name. Another bug exists here, however, which is that the buffer is never freed. The virus retrieves the current directory, and compares the first four bytes with 'WIN' and 'win'. The virus author intended to avoid the '%windir%' directory, which is by default 'WINNT' on *Windows NT* and *Windows 2000*, and 'WINDOWS' for all other platforms. However, there is a bug in the comparison: by comparing four bytes against a three-byte string, the only names that can match are 'WIN' and 'win'.

The virus searches within the current directory for all files with the '.exe' suffix. For each file that is found, the virus opens it and reads the entire file into memory, regardless of how large it is. The virus searches within the entire file for the 'msco' string. This is intended to match 'mscorlib.dll' and similar strings, to avoid the infection of *Microsoft .NET* framework files. There are simpler ways to detect such files, of course, such as the presence of the CLR Runtime Header Data Directory.

Only now does the virus check for the 'MZ' and 'PE' signatures within the file. Another bug exists here, which is that the 'PE' signature comparison is incomplete. The true signature is four bytes long, but the virus checks for only the first two bytes. While it is unlikely that any DOS programs contain such a signature, the possibility exists, and the virus might attempt to infect one as a result.

CHECKS AND BALANCES

The virus performs some simple checks to see if the file can be infected, however these checks are insufficient. The virus checks if the virtual size of the entrypoint section is zero, and if there is sufficient space to add a new section header. The file will not be infected if either of these checks fail. In the case of a section with a virtual size of zero, the physical size should be used instead – perhaps the virus author was not aware of this.

The virus does not check for 64-bit format files. As a result, such files will be infected, but incorrectly. The structure is not damaged, but the virus calculates some absolute addresses using the ImageBase field in the PE header, and in 64-bit files the field is 64 bits large and begins four bytes earlier. The result is that the top four bytes of the ImageBase are referenced instead.

The check for sufficient space for the section header is not quite correct either. The virus author intended to check whether there are 40 individual bytes available, but the virus compares four bytes at a time while advancing one byte at a time. The result is a check for three bytes more than is required.

If there is sufficient space to add a new section header, then the virus appends a new section to the file, and changes the original section names to 'UPXn', where 'n' is an increasing single-digit number. However, the new section is always named 'UPX0'. The virus fills with 'FF' values any remaining space after the new section header. This serves as the infection marker, since now it will appear that there is no space for another section.

REALLY 'NO EXECUTE'

The virus replaces completely the characteristics for the entrypoint section. It changes them to read/write/init, and does the same for the newly added section. This act is not compatible with DEP, since without the Executable flag set in the section header, the contents of the sections cannot be executed on platforms that support DEP.

The new section header states that the section begins immediately after the last section in the file. The virus checks for data that are appended outside of the image, but the check is for the presence of at least 10,000 bytes. Anything smaller than that, such as debug information, will be ignored by the virus. The virus makes some adjustments to the PE header, then writes the entire file back to the disk. At this point, the virus seeks the location of the host entrypoint and writes the first decryptor.

The virus allocates a buffer to hold a copy of the virus body, then copies itself to the buffer and encrypts the copy. Yet another bug exists here, which is that a buffer is allocated for each file to infect, but it is never freed. The result can be a very large allocation of memory if there are a lot of files.

The virus then seeks to the end of the file and writes the encrypted virus body. If there are appended data after the original last section, then they will be 'sandwiched' between the image and the virus body.

The infection is now complete, and the virus searches for another file and repeats the infection process. Once all files have been examined, the virus steps up one directory level and repeats the process, including the stepping, twice. Thus, the virus infects the current directory and two directories above it.

REMOVERS AND SHAKERS

After infecting the nearby files, the virus allocates several buffers in preparation for the final stage. As before, these buffers are never freed. The virus retrieves the list of drive letters. It skips the 'A:' drive, then looks for drive letters that represent removable media. There is a bug in the enumeration, which is that to determine when the end of the list is reached, the virus checks one byte after the current position. The correct behaviour is to check the byte in the current position, because if a debug heap is active, one byte after the current position is the beginning of the 'BAADFOOD' sequence, which continues to the end of the buffer.

If a removable drive is found, then the virus copies itself to the root directory of the drive, as 'driver_setup.exe'. After a short delay, the virus creates an 'autorun.ini' in the root directory of the drive, which contains a reference to the 'driver_setup.exe'.

CLIP GO THE SHEARS

The virus queries the clipboard for objects that are in the process of being copied. For each such object, the virus queries its file attributes. A bug exists here, which is that the virus author did not seem to realize that attributes can be combined. The virus wants to avoid directories and read-only files, but this is only successful if only those attributes are set. However, if a file has read-only and system attributes set, for example, or if a directory is hidden, then the virus will assume that both of those cases describe files.

If a file is found, then the virus will remove the filename from the path to produce a directory. If a directory is found, then the virus will call the infection and stepping routine to infect the nearby files.

The clipboard query returns the number of objects on the clipboard, and the virus is aware of this, but there is some missing code which would be used to enumerate these objects. Instead, only the first object is examined, and a value is left on the stack.

After the virus examines the first object on the clipboard, the virus repeats the dropper and clipboard procedures, beginning with the retrieval of the drive letters. However, a value is left on the stack each time that part of the code is reached, eventually resulting in a stack fault. This causes the virus to crash and the application to be terminated.

CONCLUSION

The virus author wanted to call this virus 'Divinorum', which means 'diviner', someone who can predict something about the future. Here's my prediction: your talents will be recognized and suitably rewarded.

Now read that again.

MALWARE ANALYSIS 2

ALL YOUR MP3s ARE BELONG TO US

Christoph Alme, Micha Pekrul, Dennis Elser
Secure Computing Corporation, Germany

Shortly following the appearance of the crafted malicious ASF (Advanced Systems Format) media files that hit the road earlier this year [1], a new class of malicious ASF media file appeared with growing prevalence. The malicious files seemed to have originated from legitimate (benign) media files which were all tampered with in the same way. It took another month and some more close collaboration with fellow malware researchers to finally get hold of the nasty newbie and find the missing part of the puzzle. In this article, we walk through the complete attack process of the Win32.ASF-Hijacker.A trojan, which has two different sets of victims: those affected by the trojan itself and those affected by media files altered by the trojan.

DELIVERING THE GOODS

The root of all evil lies on a 'warez' website in this instance. People searching for serial numbers, cracks or key generators may find themselves with a trojan downloader instead of 'just' the illegal content they were looking to download. In this case, the search for an activation key for the popular *Nero Burning ROM* program does the undesired trick. A simple hidden IFrame pointing to a page that embeds the executable as an OBJECT element triggers a drive-by download.

The downloader retrieves an XOR-obfuscated list of malware download URLs from 'smart-security.biz', then downloads the various malware files – one of which is the ASF-infector component. Paying respect to desktop firewalls in its very own way, the downloader first creates

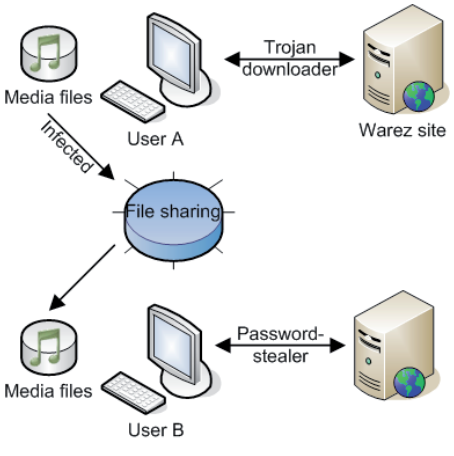


Figure 1: Attack process in two stages.

remote threads in either the *Internet Explorer* or *Firefox* browser and then lets this trusted process perform the download. To add to our headache, the downloader retrieves what appears to be a legitimate GIF image. The image can be displayed as normal, but the file's header denotes the image size as 52 x 34 pixels – GIF images of this size are normally about 2 KB in size, but this one is 36 KB, so there seems to be some 'added value' in it. A comparison of the supposed GIF image to a set of benign GIF images of the same file size reveals only slight deviations in their data's overall information entropy gradient (see Figure 2). The benign GIFs' data is just as random (e.g. compressed at similar quality), but the insidious GIF shows two anomalous downward spikes that bear some resemblance to an executable's section padding areas.

Indeed the infector component is compressed with the PECompact runtime packer, XOR-obfuscated and appended to the GIF image. While the downloader retrieves the key to deobfuscate the infector component from the file at eight bytes from its end, an x-ray scan for a valid PE header easily reveals the embedded executable as well.

Probably as a countermeasure against desktop behaviour blockers and on-disk scanning, the infector component does not touch down on the hard disk at any time – except for infecting all your precious MP3s and videos, that is. Having deobfuscated the infector component, the trojan downloader does not drop and execute it; instead it executes the legitimate 'winver.exe' from the *Windows* installation as a suspended process. Then it replaces the complete executable image in memory by unloading winver.exe's image via ntdll's ZwUnmapViewOfSection() API (from user mode), mapping the infector component into the process space through WriteProcessMemory(), and finally resuming the main thread of the process.

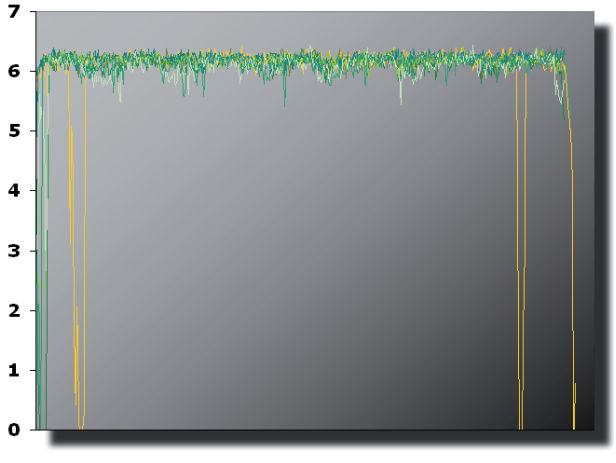


Figure 2: Information entropy gradients of insidious GIF (yellow) compared to several benign GIFs (all green).

HIJACK AND EXIT

The trojan searches for MP2, MP3 (MPEG-1 Layer 3) and ASF media files on the local hard disk and on mapped shares (all drives with a Win32 API drive type of either DRIVE_FIXED or DRIVE_REMOTE). The Windows Media Format SDK's WMCreateReader() and WMCreateWriter() APIs are used to convert MP2 and MP3 files to ASF format first, before using the WMCreateEditor() API to create an 'IWMMetadataEditor' COM object and injecting a script command via a call to the AddScript() method of the 'IWMHeaderInfo' interface of the same COM object (see Figure 3).

The Advanced Systems Format (ASF) is a general-purpose container format for media files, used for Windows Media Video (WMV) and Windows Media Audio (WMA) files, for example. The ASF format basically consists of data chunks, each starting with a 16-byte GUID identifier followed by an unsigned 64-bit field containing the length of this chunk in bytes (including the chunk's header). The interesting chunk, which can contain script commands [2], has the GUID '30 1A FB 1E 62 0B D0 11 A3 9B 00 A0 C9 03 48 F6' (in raw byte order). Similar to the 'HREF track' in *QuickTime* movies [3], script commands are executed based on elapsed playback time. Skipping the chunk's header and a first array with type information, a second array follows with all the script commands encoded in UTF16 Little-Endian. It is not zero-terminated, but the number of characters is stored in an unsigned 16-bit field in front of each string.

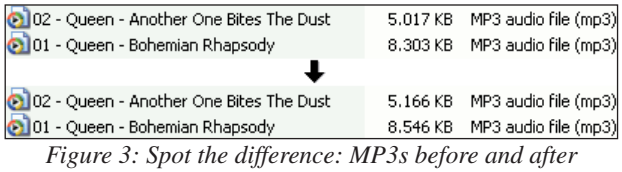


Figure 3: Spot the difference: MP3s before and after running the trojan.

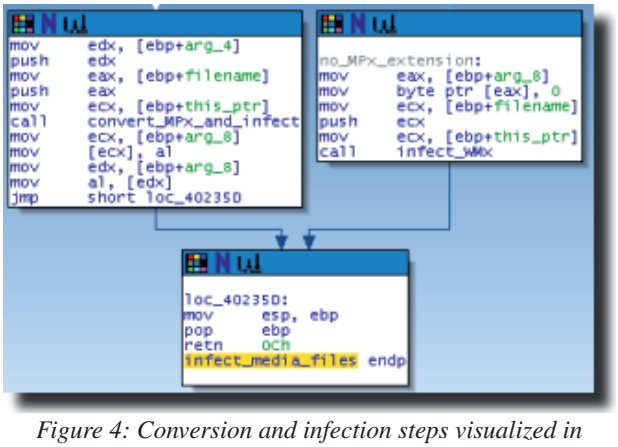


Figure 4: Conversion and infection steps visualized in IDA Pro.

The script command that the trojan injects is named 'URLANDEXIT', followed by a URL that will be opened with the default browser when playing the media file. The trojan infects all ASF media files, including the converted MP3s, with 'URLANDEXIT' commands to point collectively to 'isvbr.net', a domain hosted in Hong Kong. At the time of writing, the site returns a 302 status code to then redirect to the real malicious site, currently 'flashcodec.com'.

When done with its raid, the trojan changes a registry setting ('URLAndExitCommandsEnabled') and an INI file that alter the default behaviour of *Windows Media Player* and *WinAmp*, respectively. These changes will cause the compromised computer's media player to ignore any 'URLANDEXIT' script commands, so the user can play their videos and audio streams without noticing any change.

AND HERE COMES THE CODEC

So what's the point in infecting media files on the compromised computer, but then disabling the feature that would allow the injected script commands to run? The victim of this first stage of attack acts as a proxy to spread the second stage, and they won't notice. The user may be exchanging music files or videos on file-sharing portals or peer-to-peer networks, such as *Gnutella* (the network behind the popular *Limewire* client). As the infection remains unnoticed by the user, sooner or later they will upload files to share with others (in exchange for new files). The road to hell is paved with good intentions – sometimes illegal ones, too.

Those unlucky users who download these files and try to play them will receive a notification stating that a codec is missing and needs to be installed in order to play the video

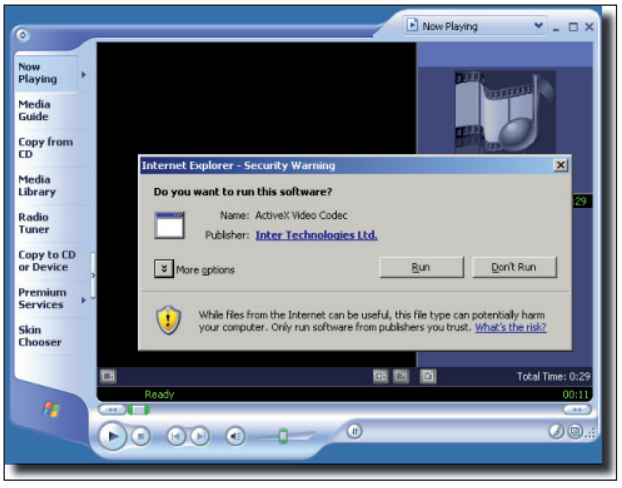


Figure 5: Do you want to run this software? No, you don't!

or audio stream. A well-known social-engineering trick, it is not surprising to find that this codec is actually malware. The only slight surprise is that in this case, it is not a Zlob trojan but an LDPinch password-stealer.

Probably in order to enforce compatibility with the *Windows Firewall* (apologies for the sarcasm), the LDPinch password-stealer adds itself to the following registry key in order to pass through it without any alerts:

HKEY_LOCAL_MACHINE\SYSTEM\...\FirewallPolicy\...\AuthorizedApplications\List

Once that has been done, it can harvest all kinds of credentials – such as from the victim's *ICQ*, *Miranda* or *Trillian* installations, credentials stored in FTP clients and in the *Firefox* and *Opera* web browsers, *Outlook* mail account credentials, and last but not least *Windows Dial-Up* network credentials. All the goods are delivered back through HTTP POST requests to 'keygenguru.com', a domain hosted in Russia.

CONCLUSION

Simply staying away from shady or illegal websites won't necessarily keep you safe these days – with SQL injection and 'malvertizing' hitting many legitimate websites – but in the case of the Win32.ASF-Hijacker.A trojan, it would. Users downloading from peer-to-peer networks need to exercise caution anyway (without going into a discussion of the legal implications), but they also need to be wary of any pop-ups that appear while playing a downloaded video or audio stream: a strong indication that they should bail out quickly.

Surprisingly, the 'missing codec' trick remains one of the most widespread and obviously successful social-engineering tricks. As is so often the case, running one's PC with a limited user account rather than as Administrator with full privileges would also help protect against the undesired installation of the fake codec. Doing so raises the bar for the attackers, and does not even require *Windows Vista*.

REFERENCES

[1] McAfee Avert Labs Blog: Fake MP3s Running Rampant. http://www.avertlabs.com/research/blog/index.php/2008/05/06/fake-mp3s-running-rampant/.

[2] Microsoft Corp.: Using Script Commands Supported by Windows Media Player. http://msdn.microsoft.com/en-us/library/aa391231(VS.85).aspx.

[3] Alme, C.; Elser, D. Blow up your video. Virus Bulletin, December 2007, p.13. http://www.virusbtn.com/virusbulletin/archive/2007/12/vb200712-blow-up-video.

CALL FOR PAPERS

VB2008 OTTAWA – CALL FOR LAST-MINUTE PAPERS

Virus Bulletin is seeking submissions from those wishing to present last-minute technical papers at VB2008, which will take place 1–3 October 2008 at the Westin Ottawa, Canada.

As usual, the conference will include a programme of 40-minute presentations running in two concurrent streams: Technical and Corporate, the running order for which has already been finalized and can be seen at http://www.virusbtn.com/conference/vb2008/programme/.

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

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

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

Those selected for the last-minute presentations will be notified 14 days prior to the conference start, and will be required to prepare a 20-minute presentation (including time for questions) to be given on the afternoon of Thursday 2 October.

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

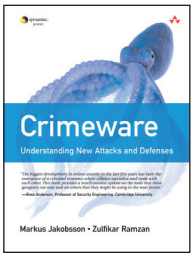
HOW TO SUBMIT A LAST-MINUTE PAPER PROPOSAL

Proposals must be sent to editor@virusbtn.com no later than **Friday 5 September 2008**. Submissions received after this date will not be considered. Please include full contact details with each submission.

BOOK REVIEW

IL BUONO, IL BRUTTO, IL CATTIVO

Paul Baccas
Sophos, UK



Title: Crimeware: Understanding New Attacks and Defenses
Authors: Markus Jakobsson and Zulfikar Ramzan (*Eds*)
Publisher: Symantec Press
ISBN 13: 978-0-321-50195-0
Pages: 608
Cover price: $54.99

Reviewing this book has been a frustrating task for two reasons: the lack of a definable structure and the gushing reviews on the covers. My review will address the first point throughout, and will be considerably less gushing.

There is a current trend among IT-related books for chapters to be written by different people on different, but related, topics. For most of these books this causes a problem with the narrative flow and they become akin to the lecture notes of a course where each class has a guest speaker. The chapters of this book are more like academic papers – in fact, some of the chapters *are* academic papers, published verbatim with seemingly no regard for whether the topic has previously been introduced. While there is a narrative thread for anyone who tries to find it, most of it is lost in the weave.

Like Sergio Leone's film this book has parts that are good, parts that are bad and parts that are ugly – and thanks to the book's format these descriptions can often be applied to different parts of the same chapter.

The move of malware authors from being electronic graffiti artists (harmless in their own minds and annoying/destructive in the minds of their victims) to serious authors of crimeware has been the pervading trend of the last decade. Defining and exploring crimeware is a laudable goal in any book, and this one starts well.

The first chapter, 'Overview of Crimeware', is a good introduction and I considered it to bode well for the rest of the book. This is followed by 'A Taxonomy of Coding Errors', which is informative, but in my opinion slightly off topic. The subject of taxonomy within the malware industry is a recurring one, however it is one I would rather leave to biologists.

Next, 'Crimeware and Peer-to-Peer Networks' comprises two research papers cleaved together into one chapter. Each part of the chapter has its own introduction, method, results and conclusion. This type of presentation is valid for undergraduate dissertations, however if I were the supervisor I would be querying the assumptions and methodology of the first part of chapter 3.

The next chapter is the one that I feel deserves most of my ire. 'Crimeware in Small Devices' contains three parts: USB, RFID and mobile. A cursory six pages are dedicated to the clear and present threats of USB device malware, three pages are dedicated to the burgeoning threat of mobile malware, while the technology which the authors say is 'on the verge of exciting times', RFID, is allocated nine pages. While the authors of the latter section (Crispo *et al.*) have done a lot of research into RFID, I do not think it deserves such a large section, or that a largely previously published piece of work warrants inclusion in such a book.

Chapters 5 and 6, 'Crimeware in Firmware' and 'Crimeware in the Browser', both have good and ugly points, the ugly being the different sections not quite jelling. The second part of chapter 5, 'Modeling WiFi Malware Epidemics', is US-centric and could have done with some examples from Europe. Chapter 6 has lots of positives even though the sections are disparate.

Chapters 7 and 8, 'Bot Networks' and 'Rootkits', are very good and along with the last two chapters provide a solid core to this book. As reference material these chapters alone are worth the price of the book. My only complaint is that the rootkit detection section is a little light.

After the high point of chapters 7 and 8 comes a low point in the form of the next two chapters, 'Virtual Worlds and Fraud' and 'Cybercrime and Politics'. These are mainly about traditional crimes rather than relating specifically to malware.

The four chapters following: 'Online Advertising Fraud'; 'Crimeware Business Models'; 'The Education Aspect of Security'; and 'Surreptitious Code and the Law', are all good. Indeed, had 'Crimeware Business Models' started the book the whole narrative might have flowed better. Chapter 14 is a little hard going for a non-lawyer and, dealing only with US law, slightly limited as a reference.

The chapter 'Crimeware and Trusted Computing' is interesting only because Trusted Computing reappears as a 'solution' to the malware/crimeware problem every few years and yet it has not yet produced a viable solution.

The penultimate chapter, 'Technical Defense Techniques', is another mishmash of subjects whose highlight is an analysis of 'Crimeware-Resistant Authentication'. The final chapter, 'The Future of Crimeware', is a good round-up of the subjects discussed.

This book has high aspirations and in parts it meets them. However, the lack of direction is problematic. A series of disparate papers joined by a common thread, the book seems more like the proceedings of a conference than a useful reference text. Nonetheless, there are several sections of the book that I will read again and I will keep it on my bookshelf for that reason.

OPINION

MALWARE TEACHING CONSIDERED HARMFUL?

Richard Ford, William H. Allen,
Florida Institute of Technology, USA

A short article in the 11 August 2008 issue of *Newsweek* magazine highlighted a 'virus-writing' class taught by Professor George Ledin at Sonoma State University in California [1]. Since then, there have been several discussions of the class online, and it has become, albeit briefly, something of a talking point in the industry. Although Ledin's stated goal is to '…teach students to think like hackers so they can devise antidotes', there is also a clear subtext in his presentation – AV vendors are not doing enough to deal with the spread of malware and are, in his opinion, obstructing independent research in anti-virus defence.

Reaction to the *Newsweek* article has been strong, with many negative comments made about the AV industry by users and IT professionals alike. Clearly, there is a public perception that industry-led anti-virus research is insufficient and that vendors do not support alternatives. Here, we will take a brief look at the history of teaching malware and whether such courses are valuable, as well as describe our own approach to teaching the subject to undergraduate and graduate students.

EVERYTHING OLD IS NEW AGAIN!

Perhaps the most important point to note is that Ledin's malware class is hardly the first of its kind. For example, in 2003, Dr John Aycock put himself firmly in the industry's cross hairs with his own 'virus-writing' class at the University of Calgary. At the time, there was significant discussion in the industry questioning the efficacy of the class and the thinking behind it, and even a public letter 'concerning the writing of viruses and how it does not teach about virus prevention' signed by more than 160 industry members [2]. Since then, many of *Virus Bulletin*'s readers have had the opportunity to meet Dr Aycock at *Virus Bulletin* conferences, and discovered him to be a reasonable man doing what he considers the right thing. Furthermore, leaving aside issues regarding perception and the legitimization of virus writing, there appears to have been no immediate damage caused by his classes or by any of his students.

Dr Aycock's class was not the first example of controversy regarding the best way to teach malware concepts. Ludwig's *Little Black Book of Computer Viruses*, the writings of the soon-disbanded ARCV, and Burger's publication of viral source code also all predate Ledin's class, and all raised very similar issues. As such, it is important to recognize that this argument has gone on for as long as viruses have been studied. Nevertheless, the topic merits re-examination given today's complex threat environment and the sometimes vocal support of these types of classes by anti-virus industry 'outsiders'.

Arguments for a teaching approach that requires students to create malware generally boil down to the assertion that this activity is an important part of the educational process – that is, that this knowledge is crucial to being able to understand malware and defend against it. To address this, educators should ask three questions in turn: 'Is it effective?', 'Is it ethical?', and 'If it is ethically questionable, are there other effective ways of teaching the subject?'. In this article, we will explore these questions based on our own first-hand experience of teaching a course on malicious mobile code at the Florida Institute of Technology.

IS IT EFFECTIVE?

It may seem strange to begin by essentially making the case for classes that require students to write malcode of various types, yet the educational value has been the mainstay of almost every argument in support of this approach (see, for example, Ledin's arguments in [3]). Furthermore, if the technique has no educational value, there is little point in taking the approach and provoking the arguments that ensue.

In terms of skill set, it is difficult to argue that virus writing is the *only* way to understand viruses. In our own course, no malware writing is involved (indeed, each student is asked to sign an ethics statement at the beginning of the semester that forbids it) and yet graduating students have a solid grasp of the fundamentals of malware analysis and defence. Stealth and hooking, for example, can easily be taught without creating something that is inherently harmful.

'To defend against viruses and other malware, one does not need to have written them.'

Researchers are often confronted with flawed analogies that show that one needs to be able to *write* viruses in order to prevent them. These arguments are beguiling at first, but rapidly fall apart. The statement 'To be able to prevent Action X one needs to have engaged in Action X…' is nonsense. Similarly, to defend against viruses and other malware, one does not need to have written them. One needs to *understand* them. The former does not require

the latter. Writing a virus may assist by showing just how simple it is, but other than that, it teaches very little.

The second argument, however, is much more complex: students learn *better* in this context. Here, effective data is lacking with respect to computer security, but is bountiful in general (see, for example [4, 5]). To the best of our knowledge, there have been no real studies that even attempt to measure the effectiveness of teaching with respect to malware. As such, it is left to the individual teacher to assess the utility of his or her teaching technique. Clearly students learn better when their interest is piqued, and the sense of being given some new and restricted knowledge can be powerful.

'Students learn better when their interest is piqued, and the sense of being given some new and restricted knowledge can be powerful.'

This may come as a surprise to those not involved in teaching at the university level, but capturing the students' attention is actually one of the key skills required by a truly effective teacher. While it would be nice to live in a world where students come to class motivated and prepared to learn, the reality is that it can require significant effort to engage a class full of students. This is made worse by the modular American university education approach, which seems to encourage compartmentalization of knowledge. Classes are frequently seen as individual units to be completed in isolation, as opposed to a more integrative world view. To the extent that virus writing excites the student, there is a fairly strong argument that knowledge will be acquired and retained more easily.

Thus, in our opinion and *without yet addressing the ethical issues*, there is likely to be some educational benefit to classes that feature the creation of live malware. Given this, we turn our attention to the question of possible ethical challenges with this approach.

IS IT ETHICAL?

The problem of ethics is that one's interpretation of 'unethical' depends greatly on the particular ethical model one follows. In order to avoid going off on tangents, for the remainder of this article we will acknowledge these cultural issues, but focus on the western educational system.

The primary ethical challenges with 'virus-writing' courses are that they:

- Pose an unnecessary risk to the health of other computers.
- Directly promote the writing of viruses by legitimizing virus writing and experimentation.

In each case, there are some reasons for concern. The most obvious objection – the inadvertent contamination of computers – is clearly a valid risk. While the risk can be reduced by strict isolation, it is impossible to eliminate it entirely. Furthermore, students may be tempted to experiment, armed with their new knowledge. Even though there may be absolutely no malicious intent, unintentional spread is clearly possible.

Similarly, many researchers have vocally objected to the idea that such efforts legitimize the work of virus writers. Once again, there is clearly an element of truth in this – the idea is that somehow an attacker is cleverly exposing a weakness in current signature-based solutions. Thus, one can argue, the virus writers are casting an unwelcome spotlight into the supposed inner workings of the industry, and are showing the truth to the masses.

As this is an argument that every serious virus researcher has heard, the danger clearly rings true. However, the facts really don't support the position: scientists are wholly aware of the limitations of signature-based products. No new malware is required to illustrate this. However, by buying into the idea, we risk once again elevating the status of the malware author from pest (or criminal) to folk hero. Developing malcode that highlights well-known weaknesses in defence is not useful.

'Developing malcode that highlights well-known weaknesses in defence is not useful.'

Finally, there is a concern that the act of virus writing makes subsequent virus/malware creation easier. Once a student has written viruses in the lab, they may be tempted to go on and become involved in virus writing. This complaint is, of course, equally applicable to classes that teach the underlying skills used by malware writers (which, frankly, should be possessed by any strong low-level programmer). As such, it is probably the weakest of the arguments against such classes.

As there is considerable debate regarding these classes, it is prudent to see whether such controversy can be sidestepped. Are there suitable alternative approaches?

A MORE EXCELLENT WAY

At Florida Tech., we have several years' experience of teaching classes related to Internet security and malicious code. At no time during these classes have we asked

students to write malcode – indeed, students are required to agree before the class not to use any of the skills they learn to do so.

The outline of the Florida Tech. course (taught at the undergraduate and graduate level) is fairly straightforward. Technical content is provided primarily by Péter Ször's (excellent) book, *The Art of Computer Virus Research and Defense*, and additional content on ethics, epidemiology, metrics and policy provided by the instructor. To date, the course has worked extremely well, with good learning outcomes (as measured by student examination performance) and no reported malware incidents.

Structurally, the course requires prior knowledge of assembly language. This is necessary, as it is vital to understanding some of the underlying properties of malware (such as how stealth actually works). A working knowledge of C/C++ and debugging is also required. Armed with this knowledge, the course first discusses legislation related to computer viruses, the university's acceptable use policy, and the ethics of malware research and defence.

From here, a historical overview of malcode is given, before following many of the topics in Ször's book. Finally, the course uses this knowledge as a foundation for exploring prevention techniques, at both the local and group level.

The primary challenge with this course is finding the appropriate way to engage students and synthesize knowledge without having to accept the risk of working with live malware. Overall, our approach has been a heavy emphasis on hands-on experimentation using *benign* programs that illustrate some of the issues typically encountered when studying malicious code. For example, it is sufficient to build a user-mode hook into the operating system to illustrate the potential for stealth – there is no need to build a virus. Similarly, we have found that simulation has provided our students with excellent insight into the techniques used to spread malware and the effectiveness of various defences.

There is certainly a good argument that our approach is more difficult than those which require the writing of attack code. However, our experience is that the primary challenge is not covering the material but presenting it in such a way that it is engaging and applicable.

PERCEPTION IS REALITY?

In summary, at least based on our own experiences in education, we believe that the challenges associated with laboratory creation of malware outweigh the possible educational benefits. However, overall it seems that these classes have had little impact, and while they may raise the blood pressure of some researchers, they are outside the scope of control. Loud complaints raise the profile of such classes, over and above their contribution to science.

Before closing, there are two side issues that bear acknowledgement. In each case, the primary issue is perception, and it is in these areas that we, as a group, are failing.

First, courses that 'teach' virus writing almost always generate significant publicity; in contrast, our quiet course has received almost no press – it is simply not newsworthy. Reading the comments on the *Newsweek* piece one gets a deep sense from the public that this sort of educational endeavour is perceived to be hugely beneficial to the student – that it will revolutionize security. In a university environment, as in business, publicity is very valuable, and the buzz surrounding this course will encourage others to follow the same path, even if the educational benefits are *de minimis*. The idea that one must know how to write viruses in order to stop them is only partly true. There is a difference between having the technical knowledge and actually exercising it. As researchers, we have clear ideas about what is possible, and this is sufficient.

Second, online discussion regarding this latest course has revealed a deep dissatisfaction with anti-malware companies. Only a fool would refuse to acknowledge that the anti-malware industry is not universally seen as the protector of computers; instead it is regarded almost as a parasite, using out-dated technologies to provide second-rate protection. Such an image is not attractive, but it is certainly reflected in the comments of many users. Spending time on a global image makeover might be a more productive use of time for members of the AV industry than fuelling the publicity behind courses that are simply a storm in a teacup.

REFERENCES

[1] Kushner, A.B. This bug man is a pest. http://www.newsweek.com/id/150465.

[2] Public letter concerning the writing of viruses & how it does not teach about virus prevention. http://www.avien.org/publicletter.htm.

[3] Ledin, G. Not teaching viruses and worms is harmful. Communications of the ACM, vol. 48, no. 1, p.144, ISSN 0001-0782, 2005.

[4] Clark, R.C.; Mayer, R.E. e-Learning and the Science of Instruction. Jossey-Bass/Pfeiffer, San Francisco, CA, 2003.

[5] Gagne, E.D.; Yekovich, C.W.; Yekovich, F.R. The Cognitive Psychology of School Learning. HarperCollins, New York, 1994.

PRODUCT REVIEW

LAVASOFT AD-AWARE 2008

John Hawes

'The Original Anti-Spyware Company', reads *Lavasoft*'s tagline, and the company's flagship *Ad-Aware* range is a grand old name with impressive brand awareness. The company is approaching its tenth anniversary, and as if to celebrate has created a spanking new edition of *Ad-Aware* with a host of upgrades and improvements.

Having started up focusing on unwanted tracking behaviour, the product has gradually branched out to cover adware, spyware and now with the integration of a full anti-malware engine, the gamut of trojans, viruses and worms, with a selection of extras thrown in for good measure. I was intrigued to see how the company's products had developed from the fairly simple on-demand-only scanners of yesteryear, to face the multi-pronged malware dangers of the modern age.

WEB PRESENCE, INFORMATION AND SUPPORT

Although many of *Lavasoft*'s users will be more familiar with finding the company's products in the 'most popular' lists of various free download sites, *Lavasoft*'s main web presence is at www.lavasoft.com. From here, localized sites in a range of languages can also be accessed.

The homepage is simple and uncluttered, adorned with soft colours and the ubiquitous photos of attractive young men and women studiously fiddling with slick laptops. Easy-to-spot buttons direct would-be downloaders or purchasers to the product range. A prominent link promises discounts to non-profit organizations, and elsewhere links lead to details of the latest threats, blog entries and industry news articles, as well as a poll.

In the company information section I discovered that, though nowadays based in Gothenburg, Sweden, *Lavasoft* was in fact set up in Germany. Alongside a selection of similar nuggets of information on the firm and its history, the company blog and industry news section keep readers up to speed in the latest developments in security, along with some jocular cartoons on office life. An awards section boasts numerous entries, mostly from download sites recognizing the popularity of the free editions over the years; the product currently boasts 200 million users worldwide.

Probing deeper into the site brought me to the full product range, which includes several versions of *Ad-Aware* itself and much more besides. The bare-bones free version has several siblings – a '*Plus*' edition, which includes the new anti-virus component as well as real-time protection; a '*Pro*' version, with even more extras and benefits, and an '*Enterprise*' version, with multiple-licence options and a central management system. A personal firewall is also available – which has scored reasonable ratings on firewall comparative sites – as well as a range of other security and system-cleaning tools: a file shredder, a registry tuner, some backup and encryption systems. All are available from the company's online shop, with the *Pro* version of *Ad-Aware* currently retailing at a fraction under $40 US. Support for several older platforms has been dropped, but the standard *Windows 2000*, *XP* and *Vista* (including 64-bit *Vista*), as well as *Server 2003*, are all covered.

Support offerings include manuals, *YouTube*-style videos and FAQs for the users of the free version. The FAQs are fairly exhaustive and most seem clear and lucid. A fairly sleepy forum is also provided to keep the free users out of trouble. Users of the paid versions are granted access to an email form for submitting technical queries. I used this form to submit a fairly tricky request in the middle of a weekday, and received a response less than ten minutes later.

A brief glance through the manual showed it to be clear, lucid and thorough. Fairly wordy but with plenty of illustrations and helpful screenshots, it clearly marks which of the functions are restricted to certain product versions, to save users of the free version from unnecessary bafflement and jealousy. Rather a lot of acronyms are used – which are all clearly explained when they first appear, but mean the manual is better suited to a full read-through rather than as a quick reference to specific sections.

The next tab on the website, the 'Security Center', provides a wealth of handy resources including the standard latest threat list, glossary of terms, research team blog and sample submission system. More peculiar to the site are the wide range of simple guides, how-tos and white papers on various aspects of security and privacy, which are written in a clear and friendly style and spiced up with quizzes, polls, interesting statistics and so on (though data sources are often vague or 'estimated' – one unlikely claim was that as many as 37% of people actually read EULAs before accepting them). One of the major subsections in this area is titled 'Threat Analysis Index'. I assumed this would be some form of malware encyclopaedia, but a quick look inside showed it to describe a risk-rating system on a sliding scale of 0–10, green to red. The name of the system was shortened to 'TAI' – an acronym I had seen sprinkled liberally throughout the manual.

INSTALLATION AND CONFIGURATION

Having managed to get hold of a full copy of the product, installation proved a pretty rapid and straightforward

process. I had a good look through the EULA, but spotted nothing untoward, and everything seemed good to go in a pretty respectable amount of time. However, once the complex licence code was entered a lengthy process of downloading and installing got under way. A few false starts were accompanied by error messages which appeared behind the splash screen, warning that the update server was busy and could not be accessed. Watching the process trundle along gave me some hints as to the source of *Lavasoft*'s new virus detection capabilities, more on which later.

Eventually things were up and running, and a clean and slick interface was presented. This provided some details on the update and licence status as well as recent scan results, with a row of options down the left-hand side. The top bar drew attention however, with its warning that real-time monitoring was not active. Some investigation showed that the 'Ad-watch' module, which provides this protection, is not active by default, but can be started either from the main GUI or from a second desktop shortcut, which opens its own separate interface. It can be set to fire up on system startup, but again this is not the default, a choice presumably influenced by the product's history as a standalone scanner commonly used in tandem with other anti-malware protection.

The system used for much of the testing was perhaps a little underpowered, towards the lower end of the requirements specified by the vendor, and as a result the flashy animation of the option bars acted rather slowly, as did some other aspects of the interface, especially the initial startup. This attempted another update each time it was opened, and I became more than familiar with the accompanying splash screen during initial exploration of the setup.

The selection of items on offer is pretty straightforward, with a scan tab providing a 'smart' scan targeting important system areas, a deeper version covering the whole machine and a customizable scan (not available in the free version). The 'Ad-watch' tab provides control over the various areas being monitored, including the registry, browsers, cookies and so on. Update and settings areas provide the obvious controls over updating and general behaviour, including the increasingly popular option to modify the appearance of the interface, with a hot pink look among the most appealing on offer. Finally, a section entitled 'Tools & plugins' provides some extra functions, which will be probed in more detail later.

SYSTEM PROTECTION AND MALWARE DETECTION

Despite the sluggishness of the interface, some initial scans on a clean system zipped through their business at impressive speed, and running for a spell with the real-time protection enabled produced no noticeable slowdown even on a tired and underpowered test machine. Moving into the lab, a system riddled with adware and more serious nasties was scanned, and everything selected picked up without difficulty. Presentation of the scan results included a selection of scary icons for the different types of threat, and a colourful bar to indicate the seriousness of the danger presented.

In another nod to the product's past in a slightly different sphere than the more traditional anti-virus products, the results are presented as a simple list of discoveries, with an empty check-box next to each. They can thus be selected for removal or left as they are, granting considerably more choice to the user than I would normally expect to see. Once each box had been checked, the process of cleaning, deleting or quarantining items was over in a flash, and highly effectively too, with all but the most innocuous registry changes and harmless files left behind as evidence of infection.

In some cases the presentation of scan results becomes a little awkward. Running some custom-designed scans over some of the *VB* test sets, I found scanning seemed to stop when it hit 5,000 items detected, with a message warning me of an exceptionally large number of infections which should be dealt with before proceeding. While this kind of number of infected items is not unusual for systems in the *VB* lab, the average user is unlikely to reach such desperate straits, at least not without seeing their machine collapse under the pressure, so the limit is perhaps justified, but it would make it a little difficult to push the product through a full VB100 comparative review.

The colourful presentation of the outcomes in the main interface only provides an infection ID and

severity indication, but a separate tab is presented at the end of the scan with more details, including a list of affected files and so on, with the empty check-boxes making an appearance to pick which items to clean up. There seemed to be no option to scroll sideways here, and with lengthy paths and more detailed detection names I found it very difficult to find much information. Looking over the more useful logs produced afterwards showed the same excellent detection results, and confirmed my suspicion that the anti-virus component has been licensed from *Avira*, whose *AntiVir* products are regular high achievers both in *VB*'s tests and those of other independent testing bodies. This engine seems a very wise choice and appears to be integrated quite seamlessly alongside *Lavasoft*'s own technology.

Another issue which might prove tricky should this product appear on the VB100 test bench any time soon, is the on-access component. Not being switched on by default after installation, and requiring specific user intervention to activate it, cause the product to fall outside the strictures of the test procedure.

The setup, like that of many products emerging from the anti-spyware sphere, does not have the option to check files on all forms of access, such as reading or writing, but instead waits for full execution before stepping in. However, this proved more than adequate to prevent infection from a random batch of samples I tried to run on the machine, which were all stopped in their tracks. Once again, an option to allow the detected files to run was presented to the user, much in the style of a firewall alert. Some brought up a selection of other alerts for suspect behaviours such as attempts to modify important sections of the registry, and in a few cases a single sample brought up half a dozen or more prompts for decisions, but none were allowed to carry out their malicious activities unchecked.

Some further exploration of the on-access configuration options dug up the ability to switch off the prompting and simply block anything judged as malicious. A contextual menu in the on-demand scan results provides the option simply to check all items, as well as a link to further data on an individual malware type. Further controls over the level of threat deemed acceptable (the 'TAI' score) and the level of heuristics active in the detection engine are also available. A command-line interface to the scanning engine is included (in the paid version), which was not tried, but the documentation suggests that it provides a good selection of options.

Measuring scanning speeds and overheads also proved a little problematic, as without full web access for activation it was not possible to get the full version running on the standard test machines, at least not without some time-consuming transferring and modification of system images. On-demand scans also had a rather long preparation period, despite the already hectic period loading the interface, and on some occasions ran to over five minutes before scanning proper started. Having got past this stage enough times, some simple tests were managed and showed pretty decent on-demand speeds, not quite as splendid as those achieved by *Avira*'s own product, but more than acceptable nevertheless. These scanning speeds were considerably slower when scanning similar numbers of infected files, as detections spark some further checking of other areas, including the registry, which may be affected by an infection. On-access overheads were less intensive than those of products that check every file as it is read or written to, but some slowdown was observed when the main interface was up and running, particularly during its frantic startup period which regularly left the rest of the system pretty sluggish for a few minutes.

During standard operation on a live system, after a little adjustment away from the default setup, the product left me feeling pretty safe in day-to-day operations, and with a scheduled scan (again not available in the free edition) set up to probe more deeply at regular intervals, *Ad-Aware* provided a solid and reliable level of security. In order to set much of this up, fairly thorough reading of the manual or built-in help may be required – unfortunately there is no contextual access to appropriate areas of the help from specific sections of the product, which would have been useful.

OTHER FUNCTIONALITY

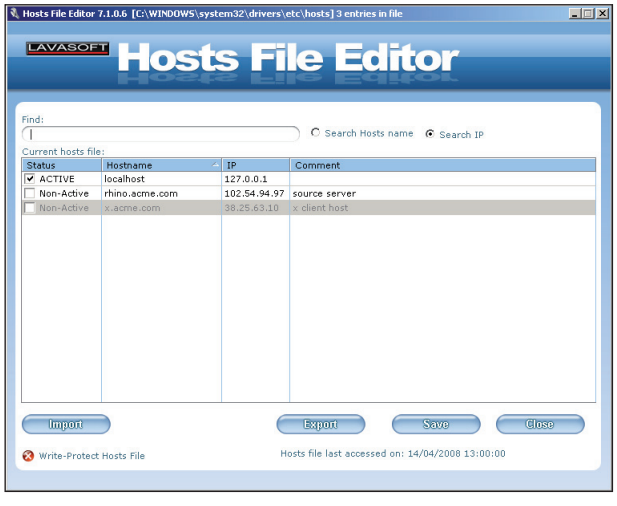
Beyond the basics of broad-spectrum protection against malware and other privacy intruders, a few extra components are included in the product. This is not a full

Internet security suite, but as the company also produces its own firewall, anti-spam and other technologies it seems likely that further development in this direction will appear fairly soon. For now, however, there are a couple of tools listed in the 'Tools & Plug-Ins' page.

The 'Process Watch' explorer tool provides a comprehensive list of running processes, along with child processes, activity and libraries in use, all with the option to terminate. This is fairly similar to the well-known *Process Explorer* tool from *Sysinternals* (now part of *Microsoft*), but presented in a slightly more friendly layout and with some less advanced options. It provides no warnings when attempting to shut down processes, and should probably be left alone by less confident users.

There is also a hosts file editing utility, which again provides a clear and shiny interface to the contents of the hosts file, and can be used by those with adequate understanding to remove unwanted entries and add new ones, to block access to specific areas for 'parental control' purposes. Again there is little guidance for those without the patience to read the manual entries, and those who already understand the use and layout of the hosts file will presumably be more than capable of making changes directly, but it is useful to have simple access to it from a central security tool.

Available from both the main interface and the secondary 'Ad-Watch' GUI is a section labelled 'Track Sweep', which monitors data stored by browsers and can be set to clear up any sensitive information, including cookies and browser history, at the end of a browsing session. This is available both as a real-time option, which by default only looks for known-bad settings, cookies and changes but can be set to erase all tracks for the paranoid, and also in the on-demand mode, to clean up as and when required. While these options are generally available in browsers themselves, once again it is handy for those with privacy concerns to have it all managed from a single point.



CONCLUSIONS

This is a pretty attractive product, combining leading detection technologies from the anti-spyware and traditional anti-virus worlds. The interface design is clear and eye-catching, if occasionally a little tardy to respond, and stability is excellent, with no errors or crashes spotted throughout the test period, other than the occasional, temporary issue connecting to the company's busy update servers.

It provides solid and reliable protection with great flexibility available to the user. Perhaps the main issue I have with it is that it does require quite some user intervention to get the most out of it. Many of the pop-ups presented provide little information that would be meaningful to the average home-user, who is relied on to make sensible decisions fairly often both when malware is detected and when genuine software (including components of the regular updates to *Windows* provided automatically by *Microsoft*) attempts to make changes to sensitive system areas. Some fine-tuning of the setup is also required to ensure maximum protection, particularly the activation of the on-access protection as standard.

This is no simple set-and-forget product, but for the more technically inclined provides some good little tools alongside excellent detection and cleaning abilities. As I have said before in these pages, a little learning is an excellent thing, and all computer users should be encouraged to take an active interest in the dangers they face on the web, but for a total novice this product would perhaps present rather a steep learning curve. For those who know what they're doing, it is certainly fit for purpose, and remains an excellent choice for what many have already long used it for: as a second layer of defence alongside another product – although the decision of which to have in place as the main on-access monitor has been made considerably less straightforward with the addition of the built-in real-time protection.

Technical details

Lavasoft Ad-Aware 2008 was variously tested on:

AMD Duron 1 GHz laptop, 256 MB RAM, running *Microsoft Windows XP Professional SP3.*

AMD K7, 500 MHz, 512 MB RAM, running *Microsoft Windows XP Professional SP2.*

Intel Pentium 4 1.6 GHz, 512 MB RAM, running *Microsoft Windows XP Professional SP3.*

AMD Athlon64 3800+ dual core, 1 GB RAM, running *Microsoft Windows XP Professional SP2* and *Windows Vista SP1* (32-bit).

END NOTES & NEWS

COSAC 2008, the 15th International Computer Security Forum, will take place 21–25 September 2008 in Naas, Republic of Ireland. For details see http://www.cosac.net/.

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.

SecTor 2008 takes place 7–8 October 2008 in Toronto, Canada. The conference is an annual IT security education event created by the founders of North American IT security usergroup TASK. For more information see http://sector.ca/.

The 3rd International Conference on Malicious and Unwanted Software (Malware '08) will be held 7–8 October 2008 in Alexandria, VA, USA. The main focus for the conference will be 'the scalability problem'. For more details see http://isiom.wssrl.org/.

Black Hat Japan 2008 takes place 7–10 October 2008 in Tokyo, Japan. Training will take place 7–8 October, with the Black Hat Briefings taking place 9–10 October. 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&xyzzy=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 representatives to discuss all aspects of electronic crime and ways to combat it. 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. This year the conference celebrates the influence of Alan Mathison Turing, British cryptographer, mathematician, logician, biologist and 'the father of modern computer science'. For full details see http://www.rsaconference.com/2008/Europe/.

Hack in the Box Security Conference 2008 takes place 27–30 October 2008 in Kuala Lumpur, Malaysia. This year's event will see new hands-on sessions designed to give attendees a closer and deeper understanding of various security issues from physical security bypass methods to the security of RFID and other wireless-based technologies. For more information see http://conference.hackinthebox.org/.

Hacker Halted Malaysia 2008 takes place 3–6 November 2008 in Selangor, Malaysia. For more information see http://www.hackerhalted.com/malaysia.

CSI 2008 takes place 15–21 November 2008 in National Harbor, MD, USA. For online registration see http://www.csiannual.com/.

The 2nd Annual Chief Security Officer Summit will take place 8–10 December 2008 in Geneva, Switzerland. For details see http://www.mistieurope.com/.

ACSAC 24 (the Applied Computer Security Associates' Annual Computer Security Conference) will be held 8–12 December 2008 in Anaheim, CA, USA. For details see http://www.acsac.org/.

AVAR 2008 will be held 10–12 December 2008 in New Delhi, India. The 11th Association of anti-Virus Asia Researchers International Conference will be hosted by *Quick Heal Technologies Pvt*. See http://www.aavar.org/avar2008/index.htm.

VB2009 will be held 23–25 September 2009 in Geneva, Switzerland. For details of sponsorship opportunities and any other queries relating to VB2009, please email conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

Editorial enquiries, subscription enquiries, orders and payments:
Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England
Tel: +44 (0)1235 555139 Fax: +44 (0)1235 531889
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

NEWS & EVENTS

ALPHABETTI SPAMGETTI

A recent paper by security researcher Richard Clayton at the University of Cambridge suggests that the volume of spam that arrives in one's inbox depends – in part – on the first letter of one's email address.

Clayton reports that addresses whose local part (left of the '@') start with an 'a' receive more spam than those that begin with a 'z'. Rather than being related to the letters' alphabetical positions, though, the discrepancy can be explained by the prevalence of those letters at the start of an address – there are more addresses starting with an 'a' than starting with a 'z'.

Clayton studied the email traffic of UK ISP *Demon Internet* over an eight-week period and found that the addresses starting with an 'a' received 35% spam in their inboxes, while those starting with a 'z' received just 20%. Other first letters that showed an increased likelihood of receiving spam included 'm' and 'p', both of which received around 42% spam, while the letter 'q' was neglected in a similar way to the letter 'z', receiving just 21%.

Clayton postulates that the most likely reason for these intriguing results is the prevalence of dictionary attacks – the more names (and addresses) there are beginning with a certain letter the more likely the spammers are to guess addresses beginning with that letter.

The findings were reported in a paper presented at the CEAS anti-spam conference last month; the full paper can be read at http://www.cl.cam.ac.uk/~rnc1/aardvark.pdf.

PRISON SENTENCE FOR PHISHER

A man who masterminded a phishing scam targeting *AOL* users over a four-year period has been found guilty of fraud and aggravated identity theft and sentenced to a total of seven years in prison, followed by three years of supervised release.

According to prosecutors Michael Dolan ran a scam in which he and five accomplices harvested thousands of *AOL* customers' email addresses, then infected victims' PCs with malicious software that would prevent them from logging on to *AOL* without entering their credit card number, bank account number and other personal information. Victims typically received a fake greetings card email containing a link which, when opened, installed the malicious package.

The defence team requested a lighter sentence, pleading mental illness on Dolan's part and arguing that fewer than 50 victims had fallen for the scam, with losses totalling around $43,000. However, according to Assistant US Attorney Edward Chang, Dolan had previously revealed that the scam had yielded $400,000 from at least 250 victims. What's more, Dolan had attempted to bribe a co-defendant, threatened to kill someone he thought was a government informant, and suborned his girlfriend to commit perjury – sounding like a fairly nasty piece of work in general.

Meanwhile, a young Lithuanian man recently managed to avoid a prison sentence in the UK after being found guilty of involvement in a phishing scam. Ronaldas Janusevicius, aged 18, was found to have been involved in a scam in which one victim received a phishing email claiming to come from *Lloyds TSB* bank. The email requested that the victim update his online banking credentials and the next he knew £9,000 had been stolen from his account.

Despite the £9,000 having ended up in Janusevicius's bank account, he claimed not to have been responsible for sending the email or obtaining the victim's account details. In fact, Janusevicius had given his own bank account details to an unknown man who had transferred the victim's money into Janusevicius's account, thus successfully covering his own tracks and leaving the young man firmly in the frame.

Luckily for Janusevicius the court acknowledged that he was not wholly responsible for the scam and ordered him to complete 100 hours of unpaid work as part of a 12-month community order, on top of paying £603 to the bank and £75 costs.

EVENTS

The 14th general meeting of the Messaging Anti-Abuse Working Group (MAAWG) will be held in Harbour Beach, FL, USA, 22–24 September 2008. See http://www.maawg.org/.

FEATURE

THE PROBLEM OF BACKSCATTER – PART 1

Terry Zink
Microsoft, USA

Have you ever received a message in your inbox informing you that an email you sent was undeliverable, and yet when you looked at the message that failed to be delivered, you saw that it contained spam – and in fact you hadn't sent it? Such notifications are known as backscatter, and backscatter is one of the hot issues in the world of spam filtering. So what exactly is backscatter? Why does it occur? How do we stop it? And why can't we do a better job of filtering it?

These are some of the questions that I will answer in this two-part article (part 2 will appear in next month's issue of *VB*).

THE LEGITIMATE CASE

Before getting into the problem of backscatter, let's look at how the system was intended to work before spammers ruined it for everyone.

Let's imagine that you want to mail a letter to your friend. You write the letter, put it in an envelope, and write your friend's address in the centre of the front of the envelope. You then put your own address on the top left-hand corner of the envelope, put a stamp on the top right-hand corner, walk to the nearest mailbox and drop the envelope in the slot. A post office representative comes, picks up the letter and then through some seemingly magical process, your friend receives the letter a few days later.

But suppose there is a problem. Let's say you write the letter to your friend and address it this way:

> Homer Simpson
> 771 Evergreen Terrace
> Springfield, USA

Aside from the fact that Homer lives at 742 Evergreen Terrace (or 743 depending on the episode), you have specified neither the state nor the zip code at which Homer lives. The post office is unable to deliver your letter so they mark it and return it to you (since you put your return address at the top of the envelope). On the letter, they put a notice such as 'Bad address' or 'Insufficient postage' or something similar. In other words, they mark the message as undeliverable.

Email works in the same way. You write an email, put your name and email address in the P1 From field (SMTP MAIL FROM) and address it to your friend, whose address you put in the P2 From field (SMTP RCPT TO). You hit 'send' in your email client and by a seemingly magical process your email is delivered to your friend in a matter of seconds.

But what happens if there is a typo in your friend's email address? Just like the post office, the email postmaster has ways of letting you know that your message was unable to be delivered. Suppose Homer did this:

```
From: Homer Simpson <hjsimpson@example.com>
To: Krusty the Klown <krustyClown@demonstration.com>
```

But Krusty's email address is actually krusty**K**lown@demonstration.com. Krusty's recipient mail server gets Homer's email, looks at the To: address and then tries to deliver the mail. But it sees that the email address doesn't exist so it sends a notification back to Homer that the message could not be delivered because the email address that he specified was invalid. This is known as a Non-Deliverable Receipt (NDR) or a Delivery Status Notification (DSN). Suffice to say, the email postmaster Homer has been sending to has been kind enough to notify him that his message did not go through. Homer gets the NDR back in his own email inbox so he can take action on it.

That's the general process of how things work.

Whose server is responsible for generating a bounce message, and under what circumstances does this occur? Once Homer's mail server has accepted the message, it must either pass it along to Krusty's mail server, or else deposit a bounce message in Homer's mailbox.

Let us say that Homer's mail server passes the message on to Krusty's mail server (at demonstration.com), which accepts the message for delivery. Unfortunately, however, a moment later the disk space on the demonstration.com server fills up, and so the mail daemon cannot deposit the message in Krusty's mailbox.

The demonstration.com mail server must then send a bounce message to hjsimpson@example.com informing Homer that his message could not be delivered to Krusty's mailbox.

Had the demonstration.com mail server known that the message would be undeliverable (for instance, if Krusty had no user account there) then it would not have accepted the message in the first place, and therefore would not have sent the bounce. Instead, it would have rejected the message with an SMTP error code. This would have left *Homer's* mail server (at example.com) the obligation to create and deliver a bounce.

So, the recipient mail server does not always generate an NDR; only in some of the cases after it has accepted the message and the SMTP conversation has finished (i.e. after the DATA command) will the recipient mail server be responsible for generating an NDR. If the reject happens

during the SMTP conversation, then it is the transmitting mail server that generates the bounce.

WHEN SHOULD BOUNCES BE SENT?

When a mail server accepts a message and later decides that it can't deliver the message, it is required to send a bounce notification to the sender of the original message. There are a few kinds of bounce notification that a mail server can send, which include:

- Recipient does not exist.
- Recipient's email inbox is full.
- Your mail server is on an IP blocklist and therefore the recipient's mail server is rejecting your mail.
- Out of office notifications (not technically NDRs but close enough).

For the first one, let's say you send a message and the recipient you are sending to does not exist; the recipient mail server lets you know by sending you a message indicating as much.

Some recipient mail servers are nice about this and they attach the original message to their email. For example, the message you get back might look something like this:

```
Date: Fri, 27 Jun 2008 06:56:00 +0000 (UTC)
From: MAILER-DAEMON (Mail Delivery System)
Subject: Undelivered Mail Returned to Sender
To: Bob's Test Account <btester@example.com>
Reporting-MTA: dns; mail111-de3.example.com
X-Postfix-Queue-ID: CCCDA18680DA
X-Postfix-Sender: rfc822; btester@example.org
Arrival-Date: Fri, 27 Jun 2008 06:55:54 +0000 (UTC)

Final-Recipient: rfc822; btester@example.com
Action: failed
Status: 5.0.0
Diagnostic-Code: X-Postfix; host winse-6216-mail4.
customer.example.com [123.21.222.101] said: 550 5.7.1
Email rejected because recipient btester@example.com
does not exist. (in reply to end of DATA command)

Subject: This is a test
From: Bob's Test Account <btester@example.com>
Date: Thu, 26 Jun 2008 23:55:49 -0700
To: Bob's Real Account <btester@example.org>

This is a test to see if I get can generate an NDR.
```

The message contains the reason the email was bounced back to Bob and the last part shows the original message (usually sent as an attachment but sometimes sent inline). Note the key characteristics of the NDR message:

- The message that arrives in your inbox is From: something that you wouldn't initially recognize like the MAILER-DAEMON, or Mail Delivery System, or something similar.
- The Subject: line of the message contains the notification that your mail could not be delivered (in this case 'Undelivered Mail Returned To Sender').
- The reason that your message could not be delivered is included in the bounce message.
- Your original message is included in the bounce message.

Here's where things get interesting. There is no set format about how NDRs should be structured. Or rather, there is no set format that everyone follows. In this case the From: address is 'Mail Delivery System', but it *could* be 'Mailer System', 'Recipient Mailer', or some other description that doesn't even contain the word mail.

The Subject: line could say 'Undelivered Mail', 'Undeliverable Mail', 'Your Message could not be Delivered', 'Delivery Status Notification', '550 Status Notice', 'We Don't Know What To Do With Your Mail', etc. The messages are usually in a foreign language if we in are Europe. In other words, this subject line could contain a variety of messages.

The reason for the bounce, like the two cases above, could be phrased in a variety of ways, in a variety of languages.

Finally, the bounced message is usually included (although this is not always the case). It could be sent inline, it could be an attachment, or it could be truncated (i.e. a partial message).

The SMTP MAIL FROM is usually a null sender, that is the SMTP MAIL FROM is <>. This is legal within the SMTP protocol. An NDR is usually sent as a null sender, but not always; sometimes they are <bounce@...> or <postmaster@...> or something similar. To make matters worse, others use null senders for non-NDR messages. This happens often with automated messages that send out reports. So, you can't count on NDRs to have null senders one hundred per cent of the time, and you can't count on null senders to be NDRs one hundred per cent of the time.

To sum up, NDRs from legitimate servers can take on a variety of forms. Everyone does it differently.

HOW DSNs SHOULD BE FORMATTED… ACCORDING TO THE RFC

There is actually a specification for how DSNs should be sent; RFC 3464 specifies the content-type for Delivery Status Notifications. This isn't an article about the RFC specification so I shall attempt to summarize it as best I can[1].

[1] RFC 3464 can be found at http://tools.ietf.org/html/rfc3464.

A DSN must provide the following:

- It must be readable by humans as well as being machine-parsable.
- It must provide enough information to allow message senders to associate a DSN unambiguously with the message that was sent and the original recipient address for which the DSN is issued.
- It must be able to preserve the reason for the success or failure of a delivery attempt in a remote messaging system, using the 'language' (mailbox addresses and status codes) of that remote system.
- It must also be able to describe the reason for the success or failure of a delivery attempt, independent of any particular human language or of the 'language' of any particular mail system.
- It must preserve enough information to allow the maintainer of a remote MTA to understand (and if possible, reproduce) the conditions that caused a delivery failure at that MTA.

FORMAT OF A DELIVERY STATUS NOTIFICATION

A DSN is a MIME message with a top-level content-type of multipart/report. In other words, it has a Content-Type header with multitype/report following it. When a multipart/report content is used to transmit a DSN:

1. The report-type parameter of the multipart/report content is 'delivery-status'.
2. The first component of the multipart/report contains a humanly readable explanation of the DSN.
3. The second component of the multipart/report is of content-type message/delivery-status.
4. If the original message or a portion of the message is to be returned to the sender, it appears as the third component of the multipart/report.

We'll see an example a little later on in this article.

The From: field of the message header of the DSN should contain the address of a human who is responsible for maintaining the mail system at the reporting MTA site, such as postmaster@.

The envelope sender address of the DSN should be chosen to ensure that no delivery status reports will be issued in response to the DSN itself, and must be chosen so that DSNs will not generate mail loops. Whenever an SMTP transaction is used to send a DSN, the MAIL FROM command must use a null return address, i.e. MAIL FROM: <>. The MAIL FROM is not to be confused with the From: field in the message headers.

At this point, I will draw to a close the summary of the proper format of an NDR. If you're any more interested in it, please read the RFC. The bottom line is we need to know that it's a bounce, the reason for the bounce and we need to see the bounce message itself. However, after having seen a lot of NDR messages, it seems to me that mail operators treat the RFC a lot like the 'Pirates' Code' from *Pirates of the Caribbean* – they're not really rules, they're more like guidelines.

SO WHAT IS BACKSCATTER?

Having worked our way through how NDRs and DSNs are supposed to work, we can now finally look at backscatter.

According to the SMTP protocol, when you send a message, you specify the HELO, the MAIL FROM, the RCPT TO, the DATA (email contents including other miscellaneous headers) and the QUIT. Here's a sample email (my comments are shown in red):

```
HELO mail.evergreenterrace.com
MAIL FROM: hjsimpson@example.com
RCPT TO: KrustyClown@example.org (this is wrong, it
should be KrustyKlown@example.org)
DATA
This is a sample message to generate an NDR message.
KrustyClown@example.org does not exist, it will
bounce.
.
QUIT
```

The message goes out from Homer's web server. The mail is routed to example.org's mail server. Rather than looking up the email address during the SMTP conversation and rejecting the message with a 5xx level error (which would force Homer's email server to send the NDR), Krusty's email server accepts the message with a 250. Later on, however, Krusty's email server sees that the email address Homer sent to doesn't exist, so it looks at the SMTP MAIL FROM, in this case hjsimpson@example.com, and sends an NDR back to Homer indicating that the message couldn't be delivered.

But, what if it wasn't Homer who sent the message? Let's say Nubar Q. Spammer decides to send a spam message to Krusty. What Nubar would do if he were a nice guy is put his email address in the SMTP MAIL FROM. But Nubar isn't about to do that. He puts Homer's email address in the SMTP MAIL FROM, because the SMTP protocol allows you to put any email address you want in the field (again, my comments appear in red):

```
HELO mail.scammers.com
MAIL FROM: hjsimpson@example.com (this is not Homer,
it is Nubar Q. Spammer forging Homer's email address)
```

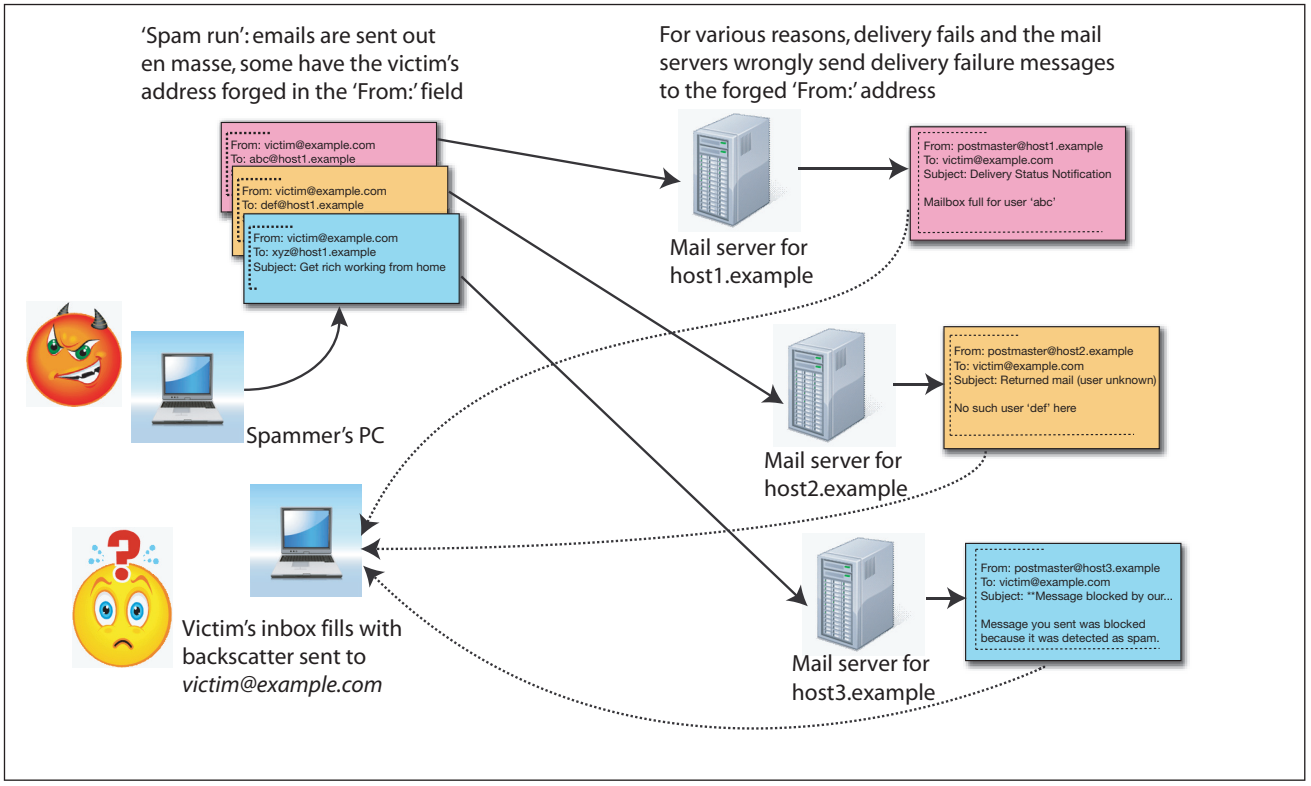


Figure 1: The problem of backscatter.

```
RCPT TO: KrustyClown@example.org
DATA
Get cheap Viagra!  Check out the following website to
get it because you obviously need it!
.
QUIT
```

Once again, Krusty's email web server accepts the message with a 250 but finds that it cannot deliver it so it looks at the email address in the SMTP MAIL FROM field. In this case, it is hjsimpson@example.com and the mail server sends an NDR 'back' to Homer.

Homer turns on his computer and opens up his email. He takes a look and sees the following in his email inbox (the first six lines below are the message headers and do not appear in the body content, and the text in red are my comments):

```
Return-Path: <> (MAIL FROM is often called the
Return-Path… note that it is null)
Date: Tue, 8 Jul 2008 22:26:56 +0000 (UTC)
From: MAILER-DAEMON (Mail Delivery System)
Subject: Undelivered Mail Returned to Sender
To: KrustyClown@example.org
Content-Type: multipart/report; report-type=delivery-
status; boundary="67A3E14185FC.1215556016/mail.
example.org" (Remember this from the Pirate's Code? I
mean RFC? It's required.)

This is the Postfix program at host mail.example.org.
I'm sorry to have to inform you that your message
could not be delivered to one or more recipients.
It's attached below. For further assistance, please
send mail to <postmaster>

- The Postfix program

(That's the human readable part)

krustyClown@example.org: host mail.example.
org[292.85.201.114] said: 550-5.1.1 The email account
that you tried to reach does not exist. (in reply to
the end of DATA command)

(That's the machine-parsable part, and it includes
the reason why it could not be delivered)

--- Below this line is a copy of the message.
Return-Path: hjsimpson@example.com
Received: (qmail 32443 invoked by uid 507); 9 Jul
2008 05:02:16 +0800
Delivered-To: krustyClown@example.org
Date: Tue, 08 Jul 2008 10:47:32 -0700
From: Homer Simpson <hjsimpson@example.com>
To: Krusty the Klown <krustyClown@example.org>
Subject: Get cheap viagra!
Content-Type: text/plain; charset=ISO-8859-1;
format=flowed
Content-Transfer-Encoding: 7bit

Get cheap Viagra! Check out the following website to
get it because you obviously need it!
```

Homer didn't send the message but it certainly looks like he did. In fact, Nubar Q. Spammer forged Homer's email address and the receiving email server sent the bounce to Homer rather than Nubar, even though Homer didn't send it. The result? Homer receives an NDR message with spam attached to it, and this entire type of spamming by proxy is known as *backscatter*.

Figure 1 illustrates the problem of backscatter on a wider scale.

Getting a single piece of backscatter is one thing, getting dozens, hundreds or even thousands of them is a major problem. While spammers may be nefarious in attempting to spam indirectly, what's more annoying is that legitimate hosts are sending piles of messages that are cluttering up inboxes and it takes forever to sort through them all.

WHY IS IT SO HARD TO BLOCK?

So why is backscatter so difficult to defend against? Here are some reasons:

1. IP reputation analysis doesn't work – spammers who spam from botnets have a weakness: public RBLs like *Spamhaus* will list them and so content filters can reject all mail from them. In the case of backscatter, the sending mail server is not a bot and is known to send good mail so it doesn't belong on a blacklist (a good blacklist, anyway).

   Finding sources of mail servers that send NDR backscatter exclusively is one thing, but if you ban all mail servers that send you backscatter, you will end up blocking a lot of legitimate mail.

2. Sender reputation doesn't work – or rather, regular sender reputation doesn't work. When most MTAs send backscatter, they usually send with an SMTP MAIL FROM as <>. This is so that if they send the NDR and their NDR bounces, the recipient MTA doesn't bounce it back again; you can't bounce to a null sender <>.

   Traditional sender reputation assumes that the inbound message is coming to you directly from another sender. So you can perform an SPF check on the SMTP MAIL FROM, but since the sender is empty, you can't verify the source of the message. The spam filter is forced to rely on something else.

3. Content filtering is more difficult – NDR messages are a pain to handle. You can't do regular sender or IP reputation analysis on NDRs in the SMTP conversation, so you have to accept the body contents of the message. Then, if you want to do the above you need to parse the body contents of the message (and not the message headers of the NDR itself) in order to extract the information you need.

   If you were to do this your spam filter would need specialized logic to recognize that the message is a bounce message and to treat it differently to extract the tokens in the message. This is trickier than it sounds because different bouncing MTAs will bounce messages differently. Remember the RFC guidelines? Hopefully, the bouncing MTA sends back the message contents including the headers. If not, or if it changes them in transit or corrupts them, header analysis is not going to work too well.

4. Regular content filtering is more difficult – if you don't want to go to the trouble of extracting many different tokens and running checks that you would normally do during the SMTP conversation (i.e. SPF or DKIM checks or IP reputation), you could default to regular content filtering. Your inbound spam filter can simply examine the message content and if there is spam, filter out the message regardless of whether or not it is an NDR.

   The problem here is that often, content filters will detect the spam and mark it as spam but they will also detect that the message is a bounce and 'de-spamify'[2] its earlier spam classification. In other words, the filter is intelligent enough to detect that this is a bounce message and possibly legitimate, thus lowering the overall spam score of the total message with spam attached. This leads to inconsistent spam filtering of backscatter. Not all of it gets through to the user's inbox, but some of it does.

SUMMARY

That summarizes the problem of backscatter. Relying on regular inbound mail filtering to detect and filter backscatter introduces problems because NDR messages are different from regular mail. They are notifications. In order to catch them we need a different method from those normally used to catch spam.

Next month, we'll look at some of the ways in which we can defend against backscatter.

[2] De-spamify is my own term derived from the words 'spam' and 'classify'. To de-spamify is to reverse the judgment of a spam filter that has previously classified a message as spam.