JULY 2006

virus BULLETIN

The International Publication on Computer Virus Prevention, Recognition and Removal

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

UNHEALTHY RELIANCE

With a lack of killer applications to spur the market, is the PC industry developing an unhealthy reliance on malicious software? Brian McWilliams presents his thoughts.

page 9

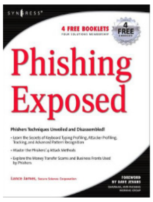
FALSE SENSE OF SECURITY

'Will the new security features in *Windows Vista* fix the virus problem?' was the question that nearly knocked Andrew Lee off his seat in surprise. Once he had recovered his composure, he decided to provide a more detailed answer than the simple, rather obvious 'no'.

page 12

vbSpam supplement

This month: anti-spam news and events; Olivier Guillion and John Graham-Cumming describe how to blind the *POPFile* spam filter with a single-word attack; David Harley asks 'How acceptable are false positives?' and reviews Lance James's new book *Phishing Exposed*; and Sorin Mustaca reports on the inaugural EU Spam Symposium.



virus BULLETIN

COMMENT

'In a perfect world, it would be nice to have no updates at all – more isn't better.'

Richard Ford
Florida Institute of Technology, USA

LESS IS MORE

As a musician, I can't count the number of times I've been told that 'less is more', and only as I have mellowed with age have I come to realize what a good maxim that is. Musical notes, cake, sunshine, even beer(!) – it's certainly possible to have too much of a good thing.

With that in mind, you can imagine my thoughts when an ad for anti-virus software arrived in the mail one morning. 'Does your current anti-virus solution update HOURLY?', the bold type read. The flyer went on to compare the number of monthly updates (600) offered by one company with that of its competitors. More, apparently, is more – at least in some people's opinion.

While I am not attacking any particular company, I think the marketing gurus who came up with the claim reflect an idea in the industry that is well past its 'sell by' date. Trying to convince consumers that more updates is better only works if we've really lost sight of the goal of anti-malcode software: *using* our computers, not *caring* for them.

Back in the day, updates were monthly 'snail mail' care packages that arrived in large 5.25-inch envelopes. Even then, there were often claims by companies who would promise to break the bondage of the update cycle. Updates were evil, or so we were told – 'Snake Oil anti-virus will detect all viruses, past, present and future' was a mantra often heard.

Somewhere between these two extremes, we've lost sight of the real goal. Anti-virus software is not a means in itself (unless you are a vendor). It's a way of making sure you get your *real* work done. In a perfect world, it would be nice to have no updates at all – more isn't better. However, in the real world it would be nice to have *fewer* updates.

Pragmatically, counting the number of updates a vendor ships is a silly way of determining how good the product is: it's a one-dimensional metric which means nothing when considered alone. More updates could mean 'latest and greatest', or it could simply mean 'heuristics so bad, we don't handle *anything* new'. Who's to say which of these is actually the case?

Aside from the lack of utility of the metric – what matters is not how many updates, but the overall level of protection provided for a particular level of customer effort – corporations must walk a treacherous tightrope between updating anti-virus solutions quickly and assuring themselves that the update itself does not cause problems in their own environment. Every change to a production system is a potential threat; balancing that against the risk of infection is difficult. Studies have shown that rolling out updates is expensive – sometimes prohibitively so. The fewer updates one needs to be safe, the better.

Underlying all of this, of course, is the tacit acceptance of anti-virus software that locks users into a *rapid* update cycle. The software versus software nature of the malcode arms race does seem to make some level of continuing update a fixture (at least for the foreseeable future), but our dependence on rapid update is a world view that we cannot accept. Relying on the network to supply updates to protect the network is a plan which is obviously flawed.

Of course, as long as customers continue to clamour for more rapid update cycles, such insanity will continue. There are ways to protect from rapid malware that are reliable, safe and don't need 600 updates a month – we need to push toward such solutions quickly. While I am not tolling the death knell for signature-based protection, stealth and speed can both throw a pretty sizeable wrench in the works. Future solutions are likely to be hybrid, borrowing the best from the old and the new.

When a methodology is flawed, doing it more doesn't make sense. However, sometimes we're all so close to the problem that we lose perspective. More really isn't better. Instead, what we need is a concerted effort to move toward solutions which actually make sense and where the user isn't eternally locked into a game of 'fast draw' with an opponent who only has to win once. Less really is more.

NEWS

NEW FACES

VB is pleased to announce the arrival of a new team member. Following the departure of Matt Ham last month, John Hawes is joining us to take over the role of Technical Consultant. John will be responsible for the all-important *Virus Bulletin* comparative reviews, standalone product reviews, prevalence data and more besides. I hope you will join me in welcoming John to *Virus Bulletin* and wishing him the best of luck - Ed.

BIG BUCKS

Anti-virus software revenue reached $4 billion worldwide last year – an increase of 13.6% on the previous year – according to industry analyst *Gartner*. The report revealed that, in 2005, enterprise and consumer sales accounted for 51.5% and 48.5% of revenue respectively, and that the top three vendors in terms of market share were (as ever) *Symantec* (53.6%), *McAfee* (18.8%) and *Trend Micro* (13.8%). *VB* waits with interest to discover what impact *Microsoft*'s entry into the anti-virus market will have on the figures for 2006/2007.

M00P GROUP MEMBERS ARRESTED

Three members of a malware-writing group were arrested last month following investigations in Finland and the UK. The three men – a 63-year-old Englishman, a 28-year-old Scotsman and a 19-year-old from Finland – who are believed to be members of the malware-writing group m00p, were arrested on suspicion of sending trojans via spam.

According to the UK's Metropolitan Police: 'This highly organised group [is] suspected of writing new computer viruses ... They have been primarily targeting UK businesses since at least 2005, and during this time thousands of computers are known to have been infected across the globe.' The three men are believed to have been involved with the distribution of the Stinx trojan.

FALSE POSITIVE REDUCTION

In *Virus Bulletin*'s June 2006 *Windows XP* comparative review (see *VB*, June 2006, p.11), *VB* reported that *Alwil*'s product *avast!* had generated three false positives during scanning of the clean test set. After discussions with the developers and further investigation of the files, *VB* now considers that the files in question were inappropriate for inclusion in the clean test set. The files have been removed and *VB* extends its apologies to *Alwil*. With a faultless performance across the In the Wild test sets and an admirable performance elsewhere, a VB 100% is belatedly awarded to *avast!*. No other products were affected.

Prevalence Table – May 2006

Virus	Type	Incidents	Reports
Win32/Netsky	File	68,135	43.01%
Win32/Mytob	File	43,185	27.26%
Win32/Mydoom	File	18,720	11.82%
Win32/MyWife	File	14,341	9.05%
Win32/Bagle	File	4,968	3.14%
Win32/Zafi	File	2,763	1.74%
Win32/Lovgate	File	2,687	1.70%
Win32/Pate	File	828	0.52%
Win32/Bugbear	File	625	0.39%
Win32/Funlove	File	535	0.34%
Win32/Feebs	File	310	0.20%
Win32/Reatle	File	200	0.13%
Win32/Sality	File	162	0.10%
Win32/Gibe	File	143	0.09%
Win32/Mabutu	File	117	0.07%
Win32/Dumaru	File	98	0.06%
Win32/Klez	File	92	0.06%
Win32/Chir	File	87	0.05%
Win32/Maslan	File	68	0.04%
Win32/Valla	File	56	0.04%
Wonka	Script	51	0.03%
Redlof	Script	38	0.02%
Win32/Brepibot	File	37	0.02%
Win32/Mimail	File	35	0.02%
Win32/Sdbot	File	12	0.01%
Win32/Magistr	File	11	0.01%
Psyme	Script	11	0.01%
Win32/Scano	File	11	0.01%
Roor	Script	10	0.01%
Win95/Lorez	File	10	0.01%
Fortnight	Script	9	0.01%
Win32/Banwarum	File	9	0.01%
Others[1]		67	0.04%
Total		158,431	100%

[1] The Prevalence Table includes a total of 67 reports across 30 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

VIRUS ANALYSIS

TUMOURS AND POLIPS

Peter Ferrie
Symantec Security Response, USA

It seems that we have reached the stage where a parasitic virus has become a novelty. That might explain why the W32/Polip virus caught us by surprise recently – we didn't expect to see one, and we certainly didn't expect to see anything of such apparent complexity. However, looks can be deceiving.

The virus author chose the name 'Polipos', which is the Spanish word for polyp, a non-malignant growth. Perhaps the virus author wanted to suggest that the virus was harmless.

While the virus certainly was written carefully, its author was not careful enough. The virus author favoured function over form, so the code is far from optimised, but it works well enough.

EXIT, STAGE LEFT

The virus begins by checksumming itself, and branches to an exit routine if the checksum does not match the expected value. This is where we encounter the first bug. The exit routine is intended to restore the patched host bytes. It requires the VirtualProtect() API to have been retrieved from kernel32.dll – however at this point no APIs have been retrieved. The virus is aware of this possibility and checks whether the address is zero.

The bug is the fact that the address is never initialised, so it always contains a non-zero value. Additionally, the virus assumes that the host module handle has been retrieved, but again, this has not occurred yet. However, neither of these problems causes a crash, since the virus uses Structured Exception Handling to trap the errors, and simply skips restoring the bytes.

The virus then copies the host bytes into a special buffer and executes them from there. This means that if the host bytes are never restored, the virus code could be called repeatedly, as often as the patched bytes are reached.

HAPI HAPI, JOY JOY

If the checksum matches, the virus will retrieve some API addresses from kernel32.dll. The APIs are located by checksum, instead of by name.

While there is nothing new about this idea, the API resolver in this virus is aware of import forwarding. This is new code for a virus, even though the problem has long been known about and documented by virus writers. It is also a requirement for the virus to work with *Windows XP* and later, since some functions, such as GetLastError(), are forwarded into ntdll.dll as RtlGetLastWin32Error().

Function forwarding exists in all 32-bit *Windows* versions, including *Windows 9x/Me*, but the forwarded functions on those platforms are not used by the virus.

Interestingly, the checksum routine is the same 16-bit CRC32 routine that has been used by a number of viruses previously. Given the technical level of the rest of the code, this routine seems a very strange choice.

The first set of APIs that the virus retrieves are related to file management. The virus branches to the exit routine if any API is not retrievable. The second of the bugs in the virus occurs here, and it is the fact that the check for retrieving all of the APIs successfully appears only after one of the functions has already been used.

The virus then retrieves a second set of APIs, most of which are related to thread management. It branches to the exit routine if more than nine APIs are not retrievable. If that check passes, the virus makes certain assumptions about which of those APIs have been retrieved successfully.

BAD SEED

At this point, the virus calls the GetTickCount() API to initialise the random number generator. The generator is seeded further by the entry point address of the virus.

There is some unused code here, which perhaps is left over from an earlier version, since a text string suggests that this is 'version 1.2'. The code loads kernel32.dll again, even though it has been used already.

The virus then retrieves a set of APIs from user32.dll, which are related to window messaging. It branches to the exit routine if any API is not retrievable.

At this point, the virus considers itself sufficiently initialised to choose a different exit routine in the event of failure. That function supports the repair of files that have data appended to their executable image, typically application installers and self-extracting archives.

STRATA MANAGER

The repair function begins by copying the infected file to the '%temp%' directory, as 'ptf[random].tmp'. The file is checksummed and compared with the checksum that the virus carries. If the checksums do not match, the virus terminates and does not even run the host.

Otherwise, the virus restores the host bytes, as before. Additionally, the virus carries a table that contains the addresses of the cavities that the host contained, in which the virus placed some of the decryptor code. The virus erases the content of those cavities, and restores the section sizes to their original values.

The virus removes the unnamed section that contains the virus body, and moves back all of the data directories that were present. It also restores the security table if it existed previously. The virus relocates all debug and/or resource information properly, if they existed. It also rewrites the file header information to remove all traces of the added section.

The virus recalculates the SizeOfCode, SizeOfInitializedData and SizeOfUninitializedData values to place into the PE header. However, these values are used only if the SizeOfCode value was zero in the original file – which can never happen, since the virus avoids such files. Otherwise, the virus uses the values that it reserved prior to the infection.

If the PE checksum field was non-zero previously, the virus checksums the file again and compares it with the checksum of the original file that the virus carries. If they match, the virus uses that checksum, otherwise the virus uses the CheckSumMappedFile() API, if it is available, to calculate a new checksum.

The repair function is capable of returning three different result codes, one of which indicates complete success. The result is checked at the end of the function, but no action is taken. That check appears to be from older code. The result is also checked again later, and if repair was a complete success, the repaired file is executed. Once the repaired file terminates, the virus waits three seconds, then deletes the file and terminates the infected process.

NEW VERSION

The virus collects information about the operating system of the victim machine, the amount of memory present, as well as the CPU family and its capabilities. Specifically, the virus retrieves the *Windows* version number, and branches to the repair routine if it finds it is running on *Windows NT*. The virus accepts all *Windows 9x* versions (it has code devoted to the special handling required there), including *Windows Me*, and *Windows 2000* and later.

The virus calls the GlobalMemoryStatus() API to find out how much physical memory exists, and exits if it is less than 64Mb. The documentation for the API states that the size field must be set first, but this is not true, and the virus author knows it.

The virus checks the CPU flags for the presence of the CPUID instruction, and if available it uses the CPUID instruction to query the CPU family and for the presence of two recent instructions. The virus requires an *Intel 80486* or better CPU, but also requires support for the CMPXCHG8B instruction (introduced in the *Intel Pentium* 1) and CMOV instruction (introduced in the *Intel Pentium* 2). The virus branches to the repair routine if one or more of these three instructions is not available.

The virus write-enables its own module header in order to place an infection marker there if one is not present already. Since this operation is supported only on *Windows NT* and later, the virus achieves this by using the undocumented VxDCall function if it is run on *Windows 9x/Me*. If the infection marker was already present, the virus branches to the repair routine.

The virus also checks if the system is shutting down, by querying the GetSystemMetrics() API, and branches to the repair routine if so. This check is supported only by *Windows XP* and later. Conveniently, however, the return value is the same if the request is unsupported, and if the system is not shutting down. As such, it is unclear whether the virus author intended to support *Windows 2000*, or was targeting *Windows XP* and later.

If all of these checks pass, the virus retrieves from advapi32.dll a set of APIs that are related to security tokens and registry key manipulation. It branches to the repair routine if any API is not retrievable.

The virus queries the 'SCRNSAVE.EXE' value of 'HKCU\Control Panel\Desktop' key. A bug exists here that results in a handle leak if the value does not exist. The returned filename is a candidate for infection.

TERMINAL DISEASE

The virus attempts to acquire the 'SeDebugPrivilege' and 'SeCreateGlobalPrivilege' privileges. The 'SeDebugPrivilege' is required for process enumeration, while the 'SeCreateGlobalPrivilege' is required by *Terminal Services* applications in order to create a file-mapping object, which the virus uses for several purposes. This is the first known virus that is aware of *Terminal Services*.

The virus creates a file-mapping object in the global namespace, whose name is the entry point code of the host. The name is adjusted to remove all zeros. Additionally, the attributes are adjusted so that they also work on *Windows XP SP2*. Within this map, the virus creates three randomly named global namespace objects, and marks the map with the string 'JIPC' ('gypsy').

On *Windows 9x/Me*, the virus allocates memory using an undocumented flag to create a shared memory region. On *Windows 2000* and later, the memory region is already shared. The virus then copies itself into the shared memory region. This copy of the virus code is used when the virus injects itself into other processes.

The virus also acquires a security descriptor to achieve full access to objects that require ACLs. This is very uncommon – other viruses simply allow *Windows* to supply the default security descriptor, with the potential associated access limitations.

The virus then checksums the current process filename if it has been run either from a subdirectory from the following list or from within the '%ProgramFiles%' or '%SystemRoot%' directories (which might be different from the list below), regardless of the drive:

\program files

\windows

\win98

\win98se

\winxp

\win2000

\winnt

\winme

Based on that, the virus intends to check the checksum against a list of 37 special filenames. The filenames belong to network-aware applications such as *Windows Messenger*, *MSN* and *NetMeeting*. However, a bug exists here – this code is reached regardless of the execution location, so the register that should hold the checksum could hold another value, and it is possible that this value can match something in the list.

If the checksum was not found in the list, the virus enumerates the windows of the current process to see if one of them corresponds to *Windows Explorer*.

If the checksum was found, or if the current process is *Windows Explorer*, the virus retrieves from wininet.dll a set of APIs related to remote file retrieval. If any APIs cannot be retrieved, the virus 'forgets' that it found any of the APIs.

ON A TIGHT SCHEDULE

The virus uses its own thread scheduler, which works across process boundaries. The reason for this is that multiple threads will be injected into remote processes, and they must be coordinated to prevent resource conflict and to synchronise their behaviour. This appears to be the work of a professional programmer.

The scheduler begins by checking whether the filename of the current process can be found in a list carried by the virus. The list is composed of names of a large number of anti-malware products, and several other applications that are known to perform self-checking. The virus disables the file infection if any of them are found.

The virus retrieves the address of the undocumented SfcTerminateWatcherThread() API from sfc.dll. The virus uses the GetProcAddress() API because its import resolver does not support functions that are imported by ordinal only. If the current process filename is 'winlogon.exe', the virus calls the SfcTerminateWatcherThread() API to disable the System File Checker.

HOOKED ON CLASSICS

The virus then retrieves the following API addresses from kernel32.dll – it retrieves only the first five if it is running on *Windows 9x/Me*, or all of them if it is running on *Windows 2000* or later:

ExitProcess

CreateProcessA

CreateFileA

LoadLibraryExA

SearchPathA

CreateProcessW

CreateFileW

LoadLibraryExW

SearchPathW

The code in these functions is parsed, instruction by instruction, using what appears to be a home-made length disassembler engine.

At 778 bytes long, this is surely one of the largest and most inefficient assembler length disassembler engines in existence. The champion of those was published in *29A#7*, and is more functional, yet only 339 bytes long (and it can even be shortened by one byte!). However, as noted previously, the author of this virus favoured function over form, so the code is far from optimised.

The disassembler is used to copy code from the API, until five bytes have been copied, or an e8 or e9 opcode is seen. In either case, if the API address could be retrieved, then it will be hooked to point to code within the virus body.

Since the data to be modified exist in a shared memory region, the virus uses a multiprocessor-compatible method to write the required number of bytes in one pass. The hooked APIs allow the virus to infect files as they are accessed, or, in the case of ExitProcess, once the process has terminated.

After hooking the APIs, the virus queues three files for later infection. Those files are the values of the 'SCRNSAVE.EXE' registry key, '%system%\logonui.exe' and '%system%\logon.scr'.

Then the scheduler enters its idle loop. Periodically, the idle loop creates a thread that checks for the presence of a debugger. If one is found, the virus stops all activity until the debugger exits.

Additionally, the virus checksums itself to ensure that two specific routines (the scheduler and detection of *VMWare*) have not been changed. A change to either of these routines will also cause the virus to stop all activity.

TIME PASSES...

The idle loop periodically calls the routine to perform the thread injection into other processes. The injection routine enumerates all running processes within the current session if running in *Terminal Services*.

The routine ignores processes whose names are any one of the following:

savedump

dumprep

dwwin

drwtsn32

drwatson

kernel32.dll

smss

csrss

spoolsv

ctfmon

temp

It also ignores the current process. While searching, the routine attempts to detect the presence of *SoftICE* and *VMWare*. The enumeration exits if *SoftICE* is found, but due to a bug, the detection of *VMWare* does not work.

For any other process found, the routine enumerates the threads within the process, looking for threads that have been created by that process (i.e. ignoring injected threads). For each of these threads, the routine suspends the thread, then sends it a message to see if the thread wakes up. If the thread does not respond, the routine injects the virus code into the remote process and redirects execution to the injected code.

The injected code then begins the whole process again (including unpacking, which is the reason for the large size of virus – the virus carries a packed version of itself). Finally, control returns to the original code in the thread.

If no thread could be suspended, the routine attempts to create a new thread within the remote process. If that is successful, the routine injects the virus code as described above.

SAY YES TO GNUTELLA

After some time, the scheduler will start the backdoor thread. First, the backdoor checks for an active network connection. If an active connection is found, the backdoor will create a hidden window, which is used to control the network activity.

The virus then retrieves a set of APIs from ws2_32.dll, if available, and otherwise from wsock32.dll. The APIs are related to network management. The backdoor exits if any of these APIs are not retrievable.

The backdoor understands the *Gnutella* 0.6 protocol, as used by *Gnucleus* and *BearShare*, among others. It watches for the arrival of *Gnutella*-specific strings, and responds appropriately.

This is not as impressive as it sounds – the protocol is open, and the source code is available freely. However, it is significant in one way: the virus can spread through the P2P network, from a compromised machine that does not have the P2P software installed.

The *Gnutella* routine works by contacting a *Gnutella* web caching server selected at random from a list carried by the virus, and retrieving the current list of connected clients. The routine then connects to these clients, so now it will be contacted if a query is made. The routine responds to queries by offering a file called 'dmckaziejdntb'. This is the virus.

The routine keeps the current contact information in the '{1DF41E2A-DA21-0412-829E-240A8C38F7A1}' value of the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths' registry key.

Periodically, infected machines will communicate with each other, by sending a special packet. These packets all start with the string 'VPacket'. For any query that contains the string 'cmdp', a particular 'VPacket' will be sent, which will cause the virus to connect back to the sender on the specified port, and download an updated version of the virus.

INFECTIOUS GROOVES

In addition to the specific files queued for infection, the virus is interested in the subkeys in the

'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths' registry key. The default value of each subkey is queried. If the filename in the data has an extension of either '.exe' or '.scr', then the file will be considered a candidate for infection.

The infection routine begins by checking if flags in the PE header specify a file of executable type, that is neither a DLL nor a system file. Additionally, the file must be for the command-line or GUI subsystem. The file will not be infected if it appears on the same anti-malware products list that is used to avoid thread injection.

The infection marker is the presence of an unnamed section. The virus adds this section, in which it places the virus body, during infection. Files are also avoided if they contain only one section, or more than 11 sections. The virus also deletes the integrity-checking databases of several anti-virus products, if any of those files exist in the same directory as the file to be infected.

The virus breaks its code into a random number of blocks, which it places into various areas of the file, including unused space at the end of other sections, and the unnamed section that the virus added. These blocks are then encrypted using a strong algorithm. While the algorithm resembles XTEA, it can probably no longer be called XTEA, since all of the important characteristics of XTEA have been changed. Specifically, XTEA is a 64-bit block Feistel network with a 128-bit key and 64 rounds.

The sum and delta values are C6EF3720 and 9E3779B9, respectively. Polip, on the other hand, uses a 32-bit block Feistel network with a 32-bit key and only 10 rounds. Additionally, the sum and delta values have been changed to 1717E09D and 9E37F9B9, respectively. Despite this weaker encryption strength, cracking the encryption is still infeasible within a reasonable time.

The decryptor is embedded within a highly polymorphic layer, which is also spread over the file. While most of it is appended to the body in the unnamed section, some parts of the decryptor are placed at the end of executable sections in the file.

The polymorphic engine itself presents nothing really new – it supports random register assignment, dummy loops and subroutines, and dummy references to the BSS section, all of which are fairly standard these days.

However, one interesting feature relates to the dummy subroutines themselves – the engine can produce subroutines that support fastcall, stdcall, and cdecl-format parameter passing, and the routines can even operate on the parameters. The results are always discarded, though.

The key weakness in the decryptor is the linear nature of its caller – the block decryption parameters are all passed from the same subroutine, so once that subroutine is found, the parameters can be retrieved and the virus code decrypted, without any significant time penalty.

The decryptor decrypts a stub, which decrypts the rest of the code and host bytes using a 32-bit xor key. Underneath that is the packed virus body. The packing algorithm is JCALG1. Underneath the packing is another layer of 8-bit xor encryption. JCALG1 is an unusual choice. It seems that it has been used by only one other virus – W32/Fizzer – which also appeared to be the work of a professional programmer.

CONCLUSION

This virus fuelled some unpleasantness in the anti-virus community: intentional withholding of samples until detection was completed.

The first company to detect the virus claimed that it had updated its product to provide full detection of the virus at that time. However, it was almost three weeks before any other company obtained samples of the virus, and a further week before everyone had received samples. For what purpose? During that time, it was demonstrated that none of the companies had managed to provide full detection of the virus, not even the first company (which updated its detection silently when the misses were found).

In fact, this virus is trivial to detect. To untrained eyes (i.e. those of the virus author), the polymorphic layer does look very complex and difficult. However, that layer contains so many constant operations, that the real instructions are recognisable instantly once the algorithm is understood. A simple repair is also quite easy.

For the really hard-core coders out there, it requires fewer than 1,000 lines of assembler to find and decrypt the virus, and restore the host bytes, using only static analysis. No emulation or debugging tricks are required.

Perhaps now the problem has been solved once and for all, and we can all get back to other work.

W32/Polip

Aliases:	W32/Polipos-A, PE_POLIP.A, p2p-worm.win32.polip.a.
Type:	Polymorphic memory-resident file infector.
Payload:	Infects .exe and .scr files; deletes integrity-checking databases of several anti-virus products.

FEATURE

MALWARE, THE NEW DRIVER OF PC SALES

Brian McWilliams
Independent writer, USA

Cynics have sometimes suggested that the anti-virus software industry secretly provides succour to virus writers. But if you're open to such computer conspiracy theories, doesn't it make more sense that *Microsoft* and *Intel*, along with PC hardware manufacturers and retailers, benefit most from today's record rates of malware infection?

DISPOSABLE MEDIA

The notion dawned on me during a recent visit to *Best Buy* [1], a large US electronics store. I watched as a middle-aged woman used a shopping cart to roll her desktop PC in from the huge parking lot, past the bright aisles of gleaming new computers, to the customer service desk.

'It's dead', she told the man behind the counter, a member of *Best Buy*'s 'Geek Squad' repair crew. After hooking up the computer and attempting to boot it, the repairman declared that her PC was probably infected with viruses and spyware, and that it would cost $199 to fix. Moments later, the woman decided to junk the three-year-old system and buy a new notebook computer for $1,000.

Each day, this scenario is played out all over the USA, and not just in the mega-stores. *Botnay Bay Computers* [2], a privately owned PC sales and service firm serving southern New Hampshire, reports that over 70 per cent of its repair work is caused by malicious code. In about half of those cases, 'the customer decides to buy a new machine rather than pay us [$80 per hour] to try to clean it', said *Botnay* technician Bud Gardner.

Similar anecdotal reports abound. In July 2005, *The New York Times* reported that the malware menace – combined with ever-falling PC prices – has led consumers to treat computers as disposable items [3].

It should come as no surprise that malware might indirectly be a big driver of new sales. Studies by *Pew Internet* [4], *Webroot* [5], and others suggest that the majority of home PCs have some sort of malicious software infection. Even mild cases can make new systems sluggish and clog their Internet connections.

Industry analysts don't seem to be tracking malware-driven purchases closely, so no hard figures are available to back up the anecdotes. But Rob Enderle, principal analyst for the *Rob Enderle Group* [6] in California, estimates that between 30 and 50 per cent of new systems sold to consumers and small businesses are the ultimate result of malicious code infections.

It seems that the scourge of malicious software has altered some consumers' upgrade calculus. Rather than pining after the newest, fastest system so they can run demanding new killer applications, many computer shoppers today just want to start over with a clean slate (or at least a pest-free *Microsoft Windows* system registry).

In the past, hardware mean-time-between-failure (MTBF), the measure of a PC component's expected lifecycle, might have dictated some system replacements. But today, sales increasingly seem to be driven by high failure rates for software – specifically, the fragility of *Microsoft*'s operating systems and its *Internet Explorer* web browser in the face of onslaughts from viruses, worms and adware.

GOOD FOR BUSINESS?

Of course, if you practice 'safe hex', you can easily keep a PC in service for many years. Case in point: the *Dell Dimension* desktop I'm using to write this article. The *Pentium 3* system is pushing seven years old and yet is still perfectly adequate for the tasks I put it through.

The PC industry can't afford too many customers like me. Instead, they'd probably prefer a marketplace full of people like the ones Lawrence Baldwin regularly assists. Baldwin runs an online intrusion monitoring service called *MyNetWatchman* [7]. His company is often hired by Internet service providers to provide remote malware removal for the ISPs' infected customers.

According to Baldwin, many novice computer users self-diagnose malware infections as simply the ravages of time: 'When their systems slow down, people tell me their CPU must be worn out,' when in fact the machine may be built with cutting-edge technology that's just been hobbled by viruses and spyware. Baldwin knows of one instance in which a frustrated owner of a new *Pentium 4* system literally tossed the PC in a dumpster.

I imagine that some junior accountant at a big PC vendor might think this explosion of spyware and other malicious code is good for business. After all, PC sales took a dive during the dot-com crash. Miraculously, malware may have helped turn the industry around in recent years.

But such a view would be woefully short-sighted. Shorter PC lifecycles may spur sales of replacement machines industry-wide, but they also serve to erode the power of brands. If computers continue their slide toward becoming disposable commodities, consumers may be unwilling to pay extra for a medallion from the likes of *Dell*, *Compaq*, or *Gateway*.

The technical support problems created by malware present another downside for the PC industry, says analyst Roger Kay of *Endpoint Technologies Associates* [8]. 'The margins on a PC are so low, a single tech support call can blow away any profit on the unit', says Kay.

MINIMIZING DAMAGE

In response, PC vendors and retailers seem to be trying to minimize their damages, or even turn malware into a new revenue generator. *Gateway*, for example, states specifically that it doesn't cover damages caused by viruses in its limited or extended system warranties. *Compaq* has a similar policy, and steers customers toward what it calls its *HP Tune-up for PC Service*. For $99, this is a 'single use' service that covers 'assistance with PC performance and problem prevention,' but coverage does not include 'break/fix troubleshooting' or 'repair diagnosis'.

Dell doesn't fix malware infections under its basic warranties; it doesn't even cover what it calls 'virus-inflicted damage' in its $69 *Dell CompleteCare* accidental damage package. To receive technical support for troubleshooting and removing viruses and spyware, *Dell* customers can purchase the company's *On Call HelpDesk* service, which costs $150 for 13 months.

Many consumers may be surprised to learn about these extra charges. Suzanne Crough, a registered nurse in Rochester, New York, certainly was. Late last year, her two-year-old *Dell* desktop began freezing up and having other performance problems. When she called *Dell*'s tech support, Crough was told she probably had a virus or other malicious software infection, and that her extended warranty didn't cover it. 'I got angry. I told them I had paid extra for the warranty. I felt like I was being held captive,' said Crough. Since her daughters rely on the computer for college homework, Crough capitulated and paid *Dell* $100 to clean out her PC's malware using remote administration.

To make matters worse, two weeks later, Crough's Internet service provider, *Time Warner Cable*, notified her that her PC was being used as a spam zombie and threatened to cut off service if she didn't get the problem corrected. Fortunately, *Time Warner* arranged to have *MyNetWatchman* handle the system cleaning at no expense to Crough.

Dell and other manufacturers have claimed that spyware-related tech support is eating into their profit margins, so I'm not going to suggest that PC vendors or retailers are getting rich from such 'repair' revenue. And, even if they were, 'if you can turn a cost into an advantage, that's good business,' said Enderle.

But what about *Microsoft*? The company stands to rack up operating-system licence royalties every time a malware-infected PC is prematurely put out to pasture and replaced with a new one. To keep this revenue stream flowing, is it possible that managers in Redmond are looking the other way on spyware and other malicious code? Could this be why, for example, *Internet Explorer*'s dangerously spyware-friendly 'install on demand' and 'browser helper objects' features are turned on by default?

No way, says Ben Edelman, a Harvard researcher who closely follows the spyware industry [9]. While the big company may not be monolithic in its view of the problem, '*Microsoft* really seems to want to stop spyware,' he said. 'For *Microsoft*, spyware poses a special problem. It harms the *Microsoft* brand, encouraging users to switch to Macs, infuriating IT administrators, and so on,' said Edelman.

As proof that *Microsoft* is moving aggressively against spyware, Edelman points to the firm's 2004 acquisition of *GIANT Company Software, Inc.*, which enabled *Microsoft* to incorporate the *GIANT* anti-spyware technology into its free spyware removal utility, now branded *Windows Defender*.

In early 2005, *Microsoft* also released a free, malicious software removal tool. Today, it is capable of cleaning dozens of worms and viruses. Yet the tool, like *Windows Defender*, doesn't run under *Windows 98*, *ME*, or *NT* – leading one to wonder whether some *Microsoft* managers view its anti-malware utilities as an upgrade trick. 'They're not as concerned about the situation as they should be, because they can increase sales because of it. I don't think it's being dealt with aggressively anywhere along the line,' said *Botnay Bay*'s Gardner.

Edelman notes that one of *Microsoft*'s fiercest weapons – its legal team – doesn't have its usual vociferousness when it comes to spyware. He says many high-tech firms seem to be paralysed by legal uncertainty. 'They fear that a spyware vendor must be good or legitimate merely because they have a licence agreement and some lawyers,' said Edelman.

Microsoft further undercut its image as a spyware warrior in July 2005, when rumours swirled that it was close to acquiring *Claria*, formerly *Gator Corporation*, a notorious adware firm. Around the same time, *Microsoft*'s anti-spyware program received an update under which the threat from *Claria*'s software and that of several other adware firms was downgraded from 'quarantine' to 'ignore'. Even though the *Claria* deal never materialized, many observers interpreted this incident as a telling sign of *Microsoft*'s ambivalence about spyware [10].

PESSIMISTIC

Whatever the motivations of the PC industry's big players, the malware situation is making consumers like Crough increasingly pessimistic. 'It seems like I'm buying a new computer every couple of years. I don’t blame *Dell* for that. It’s just the nature of the beast of computers. There’s always going to be these virus idiots who are going to do what they want to do. I don’t think there’s ever going to be a resolution of this,’ said Crough.

Kay, of *Endpoint Technologies,* says no one in the PC industry, including *Microsoft*, stands to gain long term when consumers and businesses perceive computers as unsafe or unreliable. ‘That inhibits ecommerce, which is one of the big engines driving PC shipments,’ said Kay.

According to Edelman, rather than relying on malware-driven replacements, PC vendors are better off growing the market; for example, selling additional computers to households that already have one. But he says market forces won’t be kind to vendors that can’t deliver a robust, malware-free computing experience to consumers. Edelman predicts a replay of the situation faced by the US auto industry in the 1980s, when foreign manufacturers put great pressure on American car makers to improve quality.

In the long run, said Edelman, consumers reasonably expect – and ought to receive – computers that generally work as expected. ‘The PC industry does its best to sell products that are useful, reliable, and robust – products that users actually want, and are prepared to pay good money to get,’ he said.

Here’s hoping *Microsoft* and the gang can quickly get out ahead of the malware problem – and put those crazy conspiracy theories to rest.

REFERENCES

[1] http://www.bestbuy.com/.

[2] http://www.botnaybay.com/.

[3] ‘Corrupted PC’s Find New Home in the Dumpster’, *New York Times*, July 17, 2005. http://www.nytimes.com/2005/07/17/technology/17spy.html.

[4] ‘Spyware: The threat of unwanted software programs is changing the way people use the Internet’, July 6, 2005, http://www.pewinternet.org/PPF/r/160/report_display.asp.

[5] Webroot State of Spyware report, May 11, 2005, http://www.webroot.com/stateofspyware/.

[6] http://www.enderlegroup.com/.

[7] http://www.mynetwatchman.com/.

[8] http://www.ndpta.com/.

[9] http://www.benedelman.org/.

[10] ‘Why Microsoft AntiSpyware is Untrustworthy’, *eWeek.com*, July 12, 2005, http://www.eweek.com/article2/0,1895,1836008,00.asp.

OPINION

FIXING ‘THE VIRUS PROBLEM’?

Andrew Lee
ESET, UK

Recently I was asked: ‘Will the new security measures in *Windows Vista* fix the virus problem?’ After I had recovered my seat, having almost fallen out of it with surprise, I attempted what I hope was a reasonable answer. However, having had more time to think about it, I have decided that the question really does deserve more than the simple, rather obvious answer ‘No’.

A second, also obvious, answer might be that *Microsoft* itself clearly does not believe that the new security controls will solve the problem, as it has invested large buckets of cash in developing its own anti-malware solutions (both anti-virus and anti-spyware). These are bundled with a number of other tools as *Microsoft One Care*, and will ship with *Vista*.

There are two distinct parts to the question that require investigation. Part one is: ‘What is the virus problem?’, and part two is: ‘What are the implications of the new security measures in *Windows Vista*?’ A third question then arises: ‘Does the answer to part two have an impact on the virus problem as defined in part one?’

WHAT IS THE VIRUS PROBLEM?

‘The virus problem’ is defined in popular parlance as being that bunch of ‘stuff’ that ‘causes problems on [my] PC’. Occasionally, ‘the virus problem’ is heard about in the various broadcast and other media. It includes (in the public’s perception at least) all categories of malware and a few other things besides.

This fuzzy and wide definition aside, and accepting that it will not ever be possible to satisfactorily divide out the various categories of undesired software neatly and formally (at least not in public), there is a deeper and more fundamental misunderstanding here. Not only do the general public, and even many non-specialist security people, not understand what constitutes malware and countless possible ways in which it can affect systems, but they don’t understand the application of various security controls either.

It seems that, in the minds of many, the security measures that exist do so in the main due to, and to solve, ‘the virus problem’ – they don’t, and won’t.

Access controls, user authentication, data integrity controls, cryptography, non repudiation, interception detection, service hardening and other security measures have been with us for longer than ‘the virus problem’, and do very little to address it in any meaningful way for one very good reason – they weren’t designed to.

Any sufficiently advanced operating system that is able to ‘run’ independently developed software programs is susceptible to viruses and malicious exploitation, regardless of the other types of controls (although to what extent, and to what degree of ‘usefulness’ are separate questions). Where one program can run and, for instance, collect personal information for entry into a database, another can run with the same functionality – the fact that in one case that database may be intended for the use of an attacker has no effect on the function.

The virus problem isn’t a software one, nor is it necessarily a security one, it’s not even a technology one – the virus problem is, first and foremost, a social problem. Most modern malware does not even fall into the category of what could classically be defined as ‘viruses’ – indeed the proportion is something less than 20 per cent, even if email worms are included.

So, what is it the attacker wants? Ask yourself this question: is it possible to run services and open ports on the system? If the answer is ‘yes’ (and it wouldn’t be too much use if it were ‘no’), then you will be able to control the system remotely with a backdoor program, or subvert an existing program that is listening and serving information. Is the keyboard attached to the machine for the purposes of text entry, including the entry of sensitive personal material? Again, yes, therefore keyloggers will still be a threat. Is it possible to install files onto the system? Yes? Then ‘the virus problem’ is still with us.

Any operating system that is deployed or used in an unsafe manner is subject to malicious exploitation, either directly by individuals, or by malicious software, including viruses and worms.

WHAT ARE THE NEW SECURITY FEATURES IN WINDOWS VISTA?

A list of the main ‘new’ security features in *Vista* includes: User Account Control, Consent and Credentials, Code Integrity, Data Encryption, Application Isolation, Data Redirection, Cryptography, Credential Providers, Service Hardening, *Windows Defender* and Rights Management Services [1].

Some of these we can discard, as they have nothing to do with ‘the virus problem’, and we can focus on the ones that may have an impact: User Account Control, Consent and

Credentials, Code Integrity, Application Isolation, Service Hardening and *Windows Defender*.

USER ACCOUNT CONTROLS (UAC)

This feature (long a staple of *nix and Mac OS operating systems), is not really new; it has been present for a long time within the *NT* family of operating systems. The difference is that now the user is made more aware of it, and what before was the 'run as' function is more closely integrated, in that it does not require the user to use it explicitly (assuming they were not running as an administrator anyway). Users will still be able to do important things like entering WEP keys, installing printers and running programs they've downloaded from the Internet – the only difference will be that *Windows* will ask them for permission first.

Not allowing full administrative privileges to a standard user (at least by default), if nothing else, finally brings *Microsoft*'s *Windows* OS to a point where its configuration may alert the more wary user to the problems that are common to undesirable programs. However, it does not take it far beyond that, and it doesn't solve the real problem – that people simply don't understand how to distinguish between legitimate actions and ones likely to cause a problem. Tools that will try to do that for them exist already, and they are created and marketed by security professionals who already know that user account controls don't fix the problem.

CONSENT AND CREDENTIALS

Because of the new way that UAC is implemented, consent will be required for certain operations, in the form of the system requiring the user to input an administrative password to complete the action.

A defining feature of many undesirable programs is that, in order to install themselves, they rely on people with privilege to do something. Here, 'something' could include 'something that you did because a program asked you if you wanted to allow it and you didn't know what to do, so you clicked "OK" and allowed it'.

The user is often also the administrator (especially in a large percentage of home systems), and forcing him to enter a password when the operating system asks if it can perform a function is not going to solve the problem. This is already the classical problem with some software firewalls, and it will be a problem with *Vista* for the same reason: click first, think later.

What the implementation of UAC and C&C may do is alert users to the fact that something unexpected has happened – for instance, the fact that a program attempted to install itself at startup – which may cause them to question or prevent the action.

However, this does put the onus on the user, and does require that they know what they are doing. Often inherent in assumptions about user account control is the idea that an administrative user won't do something stupid. Long ago, in his book *A Short Course on Computer Viruses* [2], Fred Cohen demonstrated that controlling user privilege is no defence because it cannot prevent higher-privileged users from running code that a lower-privileged user could not. Just because you're root, it doesn't make you smart about malware.

CODE INTEGRITY

Code integrity in *Vista* means that unsigned drivers are prevented from running in kernel mode, and checks that system binaries have not been tampered with. This, at least, should go some way toward ensuring that system files don't become infected by viruses, and that kernel mode rootkits have a hard time operating.

While this may improve stability in a system under attack, it does not prevent other files from becoming infected – nor does it prevent user mode rootkits, spyware objects, worms or trojans affecting the system.

APPLICATION ISOLATION

Application isolation assures that each process will run in its own privilege level, and a system called Mandatory Integrity Control (MIC) defines 'integrity levels', so that applications running as, say, a standard user, would run at a lower authentication level than a program running with full administrative rights, and should prevent escalation of privileges.

Very exposed programs, for instance *Internet Explorer*, will run at a low authentication level, and will not be allowed to modify users' data, or any *Windows* binaries (although, this can be adjusted so that it is allowed).

Importantly, an application running at 'low' level can only write to areas of the system that are also marked 'low'. In the case of *Internet Explorer*, this location would be the Temporary Internet Files folder. If (as is now popular with some spyware) a file is running from the Temporary Internet Files location, it should not be able to modify user data – however, if the file is moved from that location (not many users store downloaded files in that location deliberately), it will execute at the level at which the user is running (and potentially the admin level if that user has permitted it, as discussed previously).

Application Isolation is essentially a 'Good Thing', and may close down some of the vulnerabilities that are currently associated with the use of *Internet Explorer*. However, it is possible to adjust the level of integrity, so it is a likely focus of exploit finders to determine a way to do this adjustment covertly. Although it reduces a certain type of risk, it does not solve 'the virus problem' – mainly because, once again, it isn't designed to do so.

One thing that has been apparent in the lifetime of *Internet Explorer* is that vulnerabilities, when they are exploited here, hit hard. Any move toward securing *IE* further is a good one, though it seems to me that doing away with ActiveX would have been far more effective.

SERVICE HARDENING

Service hardening in the *Vista* context means that system services no longer all run with ultimate privilege. (I wonder who else has frequently run taskmgr.exe from an 'at' command to gain control over rogue system services, or in order to kill any other service.) Obviously, a scheduler does not need to be running at the highest privilege level, and the service hardening allows developers to assign services different levels of privilege based on their function.

Developers should also be able to 'write-restrict' services, so that they can only have write access to objects that allow it. The biggest problem here is that, while it is possible to harden a service, and thus reduce the impact of vulnerabilities, the effectiveness of service hardening will depend on the developer using this facility correctly.

WINDOWS DEFENDER

Windows Defender is *Microsoft*'s anti-spyware program, which it purchased from *Giant* and re-badged. An examination of the effectiveness of this product is beyond the scope of this article. One excellent feature, however, is that *WD* (who else wishes they had called it *Windows Malware Defender – WMD*?) does tell the user in good detail every time a program (even a legitimate one) takes certain actions, such as writing to the registry. For the informed user, this is useful information.

What is most interesting, though, is that despite all of the other measures taken in *Vista* to preserve system integrity and reduce the attack surface for malicious exploiters, there is still a need for a standalone (albeit bundled) application which is dedicated exclusively to dealing with undesirable programs. This, more than any other indication, is tantamount to an admission that *Microsoft* does not believe that the new security controls in *Vista* are going to solve 'the virus problem'.

The fact that *Microsoft* is also now firmly in the anti-virus game with its repackaged version of *RAV*, is another tacit recognition of this fact.

SO FAR, SO GOOD ... SO WHAT?

So, what is the impact of the new security features on 'the virus problem'? *Windows Defender* will clearly have some impact – as will user access control. It may also be the case (as it was with *Windows 9x* and *Windows NT*) that, initially, a large tranche of older malware will be rendered useless on the *Vista* platform. Clearly that is a good thing, but history shows us that eventually the bad guys catch up, and soon it's business as usual in the malware creation world.

In recent years there has been a massive trend towards criminal exploitation of malware, and this has meant huge amounts of money being invested in malware development. Just as, in the laboratories of every anti-malware software vendor on the planet, there are many people scurrying around trying to get a product out that will work on *Vista*, there are as many people (maybe even more) out there who have the money to create their own infrastructure and hire malware authors with the express purpose of bringing *Vista* to its knees.

Recently we have seen direct malicious exploitations of zero-day vulnerabilities in *MS Word* and *MS Excel*, and there is no slowdown in the number of vulnerabilities being found. It is almost a certainty that in *Windows Vista* (as in any sufficiently large piece of code) there are vulnerabilities waiting to be found, or perhaps which have already been found, and are now waiting hungrily for a few bytes of exploit code.

If the end result of the laudable new measures in *Windows Vista* is that the user feels, like so many misguided GNU/*Linux* and Mac OS users, invulnerable to attack from either viruses or the plethora of other undesirable software attacks, particularly ones that employ social engineering techniques, then we will have moved backward rather than forward.

Users of any operating system have a responsibility to educate themselves about the dangers of using their systems, and the realistic possibility that, if they do not, at some point they will fall prey to an attack.

REFERENCES

[1] http://www.microsoft.com/technet/technetmag/issues/2006/05/FirstLook/default.aspx.

[2] Cohen, F. *A Short Course on Computer Viruses*. Wiley Professional Computing. 1994.

END NOTES & NEWS

The First Conference on Advances in Computer Security and Forensics (ACSF) will be held in Liverpool, UK, 13–14 July, 2006. The conference aims to draw a wide range of participants from the national and international research community as well as current practitioners within the fields of computer security and computer forensics. For details, see http://www.cms.livjm.ac.uk/acsf1/.

Secure Malaysia 2006 will be held 24–26 July 2006 in Kuala Lumpur, Malaysia. Secure Malaysia is co-hosted by National ICT Security & Emergency Response Centre (NISER).The show will be held alongside CardEx Asia and Smart Labels 2006. See http://www.protemp.com.my/.

Black Hat USA 2006 will be held 29 July to 3 August 2006 in Las Vegas, NV, USA. See http://www.blackhat.com/.

The 15th USENIX Security Symposium takes place 31 July to 4 August 2006 in Vancouver, B.C., Canada. A training programme will be followed by a technical programme, which will include refereed papers, invited talks, work-in-progress reports, panel discussions and birds-of-a-feather sessions. A workshop, entitled Hot Topics in Security (HotSec '06), will also be held in conjunction with the main conference. For more details see http://www.usenix.org/.

ECCE2006 will be held 12–14 September 2006 in Nottingham, UK. This will be the second E-Crime and Computer Evidence Conference to be held in Europe. For full details, including a call for papers, see http://www.ecce-conference.com/.

The Gartner IT Security Summit 2006 takes place 18–19 September 2006 in London, UK. For full details see http://europe.gartner.com/security/.

ISACA's eighth annual Network Security Conference takes place 18–20 September 2006 in Las Vegas, NV, USA. The conference will offer 90-minute and half-day sessions on a range of security topics including: physical security issues, web security environment, application security, hacking concepts and tools, encryption concepts and techniques, intrusion detection and prevention systems, wireless network security, database security and continuous security monitoring. For details see http://www.isaca.org/.

HITBSecConf2006 will take place 18–21 September 2006 in Kuala Lumpur. Further details and a call for papers will be announced in due course at http://www.hackinthebox.org/.

T2'06 will be held 28–29 September 2006 in Helsinki, Finland. The conference focuses on newly emerging information security research. All presentations will be technically oriented, practical and include demonstrations. See http://www.t2.fi/uutisia.en.html.

COSAC 2006, the 13th International Computer Security Symposium, takes place 1–5 October 2006 in County Kildare, Ireland. The COSAC Forum gives attendees the chance to address topics of immediate and direct relevance to their organizations and get feedback and reality-based suggestions from other practitioners facing the same types of issues, albeit in different industries or stages of evolution or political turmoil in their security programs. For details of this fully residential event see http://www.cosac.net/.

The SecureLondon Workshop will be held on 3 October 2006 in London, UK. For details see https://www.isc2.org/cgi-bin/isc2event_information.cgi.

Black Hat Japan 2006 takes place 5–6 October 2006 in Tokyo, Japan. Unlike other Black Hat events, Black Hat Japan features Briefings only. For more information see http://www.blackhat.com/.

The 16th Virus Bulletin International Conference, VB2006, will take place 11–13 October 2006 in Montréal, Canada. Email vb2006@virusbtn.com for details of sponsorship opportunities. The full programme is now available at http://www.virusbtn.com/.

RSA Conference Europe 2006 takes place 23–25 October 2006 in Nice, France. See http://2006.rsaconference.com/europe/.

Infosecurity USA will be held 24–25 October 2006 in New York, NY, USA. See http://www.infosecurityevent.com/.

AVAR 2006 will be held 4–5 December 2006 in Auckland, New Zealand. See http://www.aavar.org/.

ADVISORY BOARD

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

Corporate rates include a licence for intranet publication.

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

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

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

vb Spam supplement

CONTENTS

NEWS & EVENTS

PHONE PHISHES

A sneaky new phishing technique emerged last month, in which initial contact is made with victims via SMS text message. In the attack, an SMS message is sent to the victim, thanking them for subscribing to a (fake) online dating service. The message informs the victim that a subscription fee of $2 per day will be added automatically to their telephone bill – unless they choose to unsubscribe from the service, which can be done by visiting the (bogus) company's website. On arrival at the website, a trojan is downloaded onto the victim's machine, giving the attackers remote control of the machine and enabling them to incorporate it into a bot network.

EVENTS

The third Conference on Email and Anti-Spam, CEAS 2006, will be held 27–28 July 2006 in Mountain View, CA, USA. Full details can be found at http://www.ceas.cc/.

The Text Retrieval Conference (TREC) 2006 will be held 14–17 November 2006 at NIST in Gaithersburg, MD, USA. More details of the TREC 2006 spam track including information on how to participate can be found at: http://plg.uwaterloo.ca/~gvcormac/spam/.

FEATURE 1

BLINDING POPFILE VIA A SINGLE-WORD ATTACK

Olivier Guillion
Myriad Software, France

John Graham-Cumming
Independent consultant, France

In the four years since Paul Graham's 'A plan for spam' [1] was published, Bayesian spam filters have proven their efficiency, versatility and simplicity of operation.

Bayesian spam filters work by computing the 'spaminess value' of a message (the probability that a message is spam; typically a message with probability 1 is certain to be spam, while a message with probability 0 is certain to be legitimate). The spaminess value of a message is calculated according to the probability of each of the words in the message appearing in a spam message. These probabilities are calculated by training a Bayesian filter with samples of spam and legitimate (ham) messages.

A Bayesian filter operates by splitting the header and body of a message into tokens. These are usually words, but may include other meta-information such as the presence of HTML or attached images. The probability that the token appears in spam messages is computed for each token in the message.

After having kept only the most significant tokens (those that have a probability that is significantly different from neutral; i.e. far away from 0.5), the remaining values are combined by following the Bayes formula [2], to obtain a single probability that tells the filter whether the message should be considered as spam.

To determine the spam probability of each token, the filter is trained by the user: it has to be taught what is spam and what is not. During this training process all tokens are collected into a database specific to each user, called a corpus, that keeps track of the number of times each token has been found in each category of message (spam or ham). This 'known tokens' set, along with the spam probability of each token, differs from one user to another, so it becomes impossible for a spammer to find a single word that will be considered the same way by each of their potential victims.

How the probability for each token is computed differs from filter to filter. Filters such as *SpamBayes* [3] and *DSPAM* [4] count the number of messages in which a token appears and ignore whether the token appears once, twice, or 100 times in a single message. *POPFile* [5], on the other hand, counts the total number of tokens seen in all the training data, and the number of times a word is seen in a message including duplicates. In this article we demonstrate how this feature of *POPFile* can be exploited to get spam messages delivered.

A large number of filters were tested in the TREC 2005 Spam Track and performed well (see [6]). However, spammers have tried to poison Bayesian filters by a variety of methods. A summary of such attacks can be found in [7].

WORD SALAD AND STATISTICAL ATTACKS

To work around this form of spam filtering, spammers have tried many tricks [8], with particular focus on manipulating HTML to hide likely spammy words (e.g. 'Viagra') from spam filters.

The most common form of attack on Bayesian filters is to add to the spam messages English words chosen randomly from a dictionary. However, both [9] and [10] have shown that this actually increases the catch rate for spam because most of these words are uncommon and by picking randomly, spammers are more likely to hit words seen in spam messages than in ham (because they themselves are adding random words to spam, whereas ham messages use a more restricted vocabulary).

If a spammer had knowledge of a set of words that are likely to appear as ham in most of the corpuses of trained Bayesian filters, they would be able to conduct statistical attacks, by maximizing the probability for their messages to contain a number of ham words large enough to move the probability towards ham [11]. However, it is possible to construct an efficient attack that works against *POPFile* by choosing words that are likely to be ham for most people.

POPFILE SPECIFIC IMPLEMENTATION

POPFile computes the spaminess probability of a message in a different fashion from most spam filters:

1. It does not apply any 'windowing' to the tokens it finds in a message; all tokens are used in the probability calculation. This is based on the assumption that each token, even if it doesn't obviously belong to a given category, can be important in the final decision.
2. When a token is found multiple times in a message, the multiple occurrences are processed individually. This means that if the word 'Viagra' appears in a message ten times, *POPFile* will apply the probability for 'Viagra' ten times. This is done because *POPFile* considers that a token that appears several times is more important than one that appears only once.
3. *POPFile* maintains a 'stop word' list of common words that are ignored when processing messages. This list contains many common English words.

POPFILE SPECIFIC ATTACK

Instead of choosing words randomly from a dictionary, a spammer can maximize his chances of bypassing the spam filter by selecting very common words, that are likely to appear in ham messages. Lists of common words are readily available – see, for example, [12]. Figure 1 shows the 100 most common English words from [12], with the *POPFile* 'stop words' removed.

```
they one hot word what some other put use how said
each which their time way about many then them write
like these long make thing see two look more day come
number sound most people over know water than call
first who down side now find new work part take get
place made live where after back little only round
man year came show every good give under name very
through just sentence great think say help low line
differ turn cause much mean before move right boy old
too same tell set three want air well play end
```

Figure 1: 100 most common English words seen by POPFile.

Examination of the corpuses of a number of long-time *POPFile* users has shown that between 50% and 80% of these are hammy words (i.e. have a probability that indicates that they are more likely to appear in ham messages than in spam). For many of them, the probability is close to 0.5 – indicating that they appear often in both spam and ham, but they are, nevertheless, hammy.

POPFile counts duplicates of tokens in a message, so when a single word from the common word list is repeated several times, each of its occurrences will be processed. If the number of repetitions is large enough, even a slightly hammy word could 'blind' *POPFile* – the probability for that single token would outweigh all the other tokens in the message and force *POPFile* to classify the message into the category associated with that token (ham).

Thus, by selecting only one word from the common English word list, and repeating it in a message, the spammer can cause *POPFile* to classify their message wrongly as ham between 50% and 80% of the time.

THE ATTACK IN PRACTICE

Figure 2 shows a genuine spam message received by one of the authors and classified correctly by his *POPFile* installation as spam.

```
Return-Path: <swe4etp07@hotmail.com>
Delivered-To: olivier@guillion.net
Received: (qmail 22081 invoked from network); 9 May
2006 19:09:12 -0000
Received: from unknown (HELO guillion.net)
(195.50.152.74)
by mrelay5-1.guillion.net with SMTP; 9 May 2006
19:09:12 -0000
To: <myrasutton@guillion.net>
From: "Ernest" <aspenjazzmookie@baggy-eyed-and-
blotto.com>
Date: Mon, 15 Dec 2003 01:07:08 GMT
Subject: DISCREET OVERNIGHT PHARMACY
Content-Type: text/plain;

Pay a ton less regular price drugs

http://ffutkd.slicebred.info/?35319980
```

Figure 2: A sample spam identified correctly by POPFile.

Figure 3 shows that *POPFile* identified 27 tokens that had spam probabilities associated with them, resulting in a spam probability of 0.999829 for the message in Figure 2. *POPFile* also saw 24 hammy tokens with a ham probability of 1.713392e-004. *POPFile* considers the message to be spam.

Classification	Count	Probability
spam	27	0.999829
ham	24	1.713392e-004

Figure 3: POPFile probability calculation for spam in Figure 2.

To attack *POPFile* we chose a word randomly from the 1,000 most common English words and added it to the message 1,000 times. In this example, the chosen word was 'one'. Figure 4 shows *POPFile*'s display of the probability for the word 'one'.

Classification	Probability
spam	0.4643718161
ham	0.5356281839

Figure 4: POPFile probability display for the word 'one'.

Hence 'one' is close to being a neutral word (i.e. having a probability of 0.5), but nevertheless it is more likely to appear in ham messages than spam messages. Figure 5 shows the original spam message blinded by adding the word 'one' 1,000 times.

When examined by *POPFile*, the message is wrongly classified as ham and passes through the filter. Figure 6 shows the updated probability display from *POPFile*. *POPFile* now clearly believes that the message is ham (with a probability of 0.999999).

INCREASING THE FALSE POSITIVE RATE

By using this method against *POPFile*, a spammer can make a large number of their messages pass through the filter. A ratio of 50 to 80%, compared to the average false negative rate reported by *POPFile* users of less than 1%, is more than attractive.

```
Return-Path: <swe4etp07@hotmail.com>
Delivered-To: olivier@guillion.net
Received: (qmail 22081 invoked from network); 9 May
2006 19:09:12 -0000
Received: from unknown (HELO guillion.net)
(195.50.152.74)
by mrelay5-1.guillion.net with SMTP; 9 May 2006
19:09:12 -0000
To: <myrasutton@guillion.net>
From: "Ernest" <aspenjazzmookie@baggy-eyed-and-
blotto.com>
Date: Mon, 15 Dec 2003 01:07:08 GMT
Subject: DISCREET OVERNIGHT PHARMACY
Content-Type: text/plain;

Pay a ton less regular price drugs

http://ffutkd.slicebred.info/?35319980

one one one one one one one one one one
[followed by 99 identical lines]
```

Figure 5: A spam message blinded with the word 'one'.

Classification	Count	Probability
ham	1024	0.999999
spam	1027	5.868154e-059

Figure 6: POPFile probability calculation for spam in Figure 5.

Moreover, if the user reclassifies these messages as spam (as users are encouraged to do when spam is misclassified), the chosen word ('one' in the case above) will see its spam word count increased by 1,000, which means that, from then on, it will be likely to be considered by *POPFile* as spammy.

If the operation is performed several times (for example, if the user receives a number of spams because of this blinding technique), making enough innocent and common words become heavily spammy, the risk of regular ham messages being misclassified as spam will increase significantly.

Figure 7 shows a legitimate email that was classified initially by *POPFile* as ham. As shown in Figure 8, *POPFile* considers this to be a ham message with high probability (0.999999). To attack *POPFile* we sent three spam messages and added the words 'thanks', 'next' and 'end' to each message 1,000 times. These three words all appear in the 1,000 most common English words.

The words were chosen specifically to attack the ham message shown in Figure 7, but it is easy to see how a spammer sending hundreds of messages to the same user could choose these same words even using random selection from the list. *POPFile* was blinded by each message and classified the three spams as hams. Then each message was reclassified in *POPFile*'s user interface, teaching it that the three messages were in fact spam and not ham. Figure 9 shows the probability display for the same ham message (from Figure 7) after the three spams had been reclassified.

```
Return-Path: <info@harmonyassistant.com>
Delivered-To: olivier@guillion.net
Received: (qmail 17698 invoked from network); 31 Mar
2006 06:57:47 -0000
Received: from wp043.hoster.de (80.217.132.52)
Received: by wp043.hoster.de running Exim 4.43 using
esmtpsa (TLSv1:RC4-SHA:128)
Date: Fri, 31 Mar 2006 08:58:09 +0200
From: Kurt Stahl <info@harmonyassistant.com>
X-Mailer: The Bat! (v3.71.03) Professional
X-Priority: 3 (Normal)
Message-ID:
<513267667.20060331085809@harmonyassistant.com>
To: olivier@guillion.net
Subject: Re[2]: mail archive
In-Reply-To:
<442A4CF0.22607.25BF54@olivier.guillion.net>
References:
<104733603.20060328231629@harmonyassistant.com>
<442A4CF0.22607.25BF54@olivier.guillion.net >
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-15
Content-Transfer-Encoding: 8bit

Hi,

thanks for your mail.
I'll have a look at it next week.

Have a nice week-end!

Kurt
```

Figure 7: A ham message seen by POPFile.

Classification	Count	Probability
ham	46	0.999999
spam	42	1.650895e-008

Figure 8: POPFile probability display for message in Figure 7.

Classification	Count	Probability
spam	42	0.999829
ham	46	1.713392e-004

Figure 9: POPFile probability display for message in Figure 7 after blinding.

Hence the spammer has managed to introduce a false positive by causing *POPFile* to classify the message in Figure 7 as spam.

If the spammer repeats the added word significantly more than 1,000 times, say 10,000 times, the corpus corruption could become much more problematic. In that case reclassifying the false positive may not work; *POPFile* may still think the message is a spam.

However, that would require that the spammer be willing to add 10,000 words to a message. At an average of five letters per word the spammer has to add more than 58kB to each spam sent, increasing their bandwidth cost or time to send.

DEFENCES

Several countermeasures are possible against this attack:

1. Ignoring a wide list of common words: by adding more words to the *POPFile* stop word list, this attack could be defeated easily.
2. Capping the number of times a word is counted in a single message: any token that appears above some capping value would either be disregarded completely or capped at the maximum repetition value. Instead of using a fixed value for this bound, a reasonable solution might be to calculate it as the frequency with which the word appears in the message and exclude words that appear very frequently for an individual message.
3. Automatically ignoring words with a probability close to 0.5. Since many of the common words appear in both spam and ham with approximately equal probability, ignoring words with a probability close to 0.5 could be effective in deterring this attack.

REFERENCES

[1] Graham, P. ‘A plan for spam’. August 2002. http://www.paulgraham.com/spam.html.

[2] Graham-Cumming, J. ‘Naïve Bayesian text classification’. *Dr Dobbs Journal*. May 2005. http://www.ddj.com/184406064.

[3] SpamBayes. http://spambayes.sf.net/.

[4] DSPAM. http://dspam.nuclearelephant.com/.

[5] POPFile. http://getpopfile.org/.

[6] TREC 2005 Spam Track. http://plg.uwaterloo.ca/~gvcormac/trecspamtrack05/.

[7] Graham-Cumming, J. ‘Does Bayesian Poisoning exist?’. *Virus Bulletin*, February 2006, pp.S2–S4.

[8] Graham-Cumming, J. ‘The Spammers’ Compendium’. 2003–2006. http://www.jgc.org/tsc/.

[9] Graham-Cumming, J. ‘How to beat an adaptive spam filter’. MIT Spam Conference 2004. http://www.jgc.org/SpamConference011604.pps.

[10] Lowd, D, Meek, C. ‘Good Word Attacks on Statistical Spam Filters’. CEAS 2005. http://www.cs.washington.edu/homes/lowd/ceas05lowd.ppt.

[11] Stern, H, Mason, J, Shepherd, M. ‘A linguistics-based attack on personalised statistical e-mail classifiers’. Technical Report, Dalhousie University. March 2004. http://www.cs.dal.ca/research/techreports/2004/CS-2004-06.shtml.

[12] About.com. ‘1,000 Most Common Vocabulary Words in English’. http://esl.about.com/library/vocabulary/bl1000_list1.htm.

FEATURE 2

HAMFIGHTING – HOW ACCEPTABLE ARE FALSE POSITIVES?

David Harley
Small Blue-Green World, UK

There has been ongoing controversy over the last couple of years about the (allegedly) aggressive nature of *Verizon*'s anti-spam strategy. Complaints in various forums of poor email delivery service from the ISP seemed to be confirmed by claims from *Verizon* 'insiders' that a policy of rejecting mail by IP block resulted in the loss of all mail from large portions of Europe and Asia. This led to a much publicized class action, resulting in a settlement offer from *Verizon* to compensate customers who lost legitimate mail between October 2004 and May 2005.

But let's take a step back from the specifics of the *Verizon* case. After all, the settlement offer leaves the exact details of *Verizon*'s policy and actions unresolved and unconfirmed, and we can only speculate on the accuracy of the whistleblower comments posted in various forums and quoted widely elsewhere [1]. In any case, there are no obvious indications that *Verizon* was actually in breach of its terms of service, however aggressive its policies. The affair does, though, call into question some widely held assumptions.

100% SPAM BLOCKING

Can an anti-spam service stop all spam being delivered? Probably not, if only because there will always be a grey area where one man's spam is another man's ham. And, of course, some people have not yet developed sufficiently sensitive taste buds to distinguish between the two. Without getting tied up and bogged down in exact definitions, I guess that most of us would be happy to lose the unholy mixture of nuisances that assault our filters – kiddie porn, 419s, cheap c1al15 and v1ag4, phishes, pennystox bulletins, job opportunities in the burgeoning money-laundering market, OEM versions of our own software, tsunami victim hoaxes, religious epiphanies and all. However, there are two prevailing views of spam management:

- It isn't possible to stop all spam from being delivered.
- It may be possible to stop all spam, but only if you accept that some legitimate mail may be lost. A *Verizon* statement puts this view even more starkly: 'Any spam-blocking method will, inevitably, result in the blocking or delay of legitimate email.' [2]

In fact, these are not mutually exclusive philosophies. They're two points on a spam management continuum between allowing all mail and allowing none. Few businesses go so far as to reject all external email, though some public sector departments are absurdly bashful, and go to extreme lengths not to publicize their email addresses or web pages. One might almost suspect a conscious rate-limiting approach to workflow management.

However, it's not unusual for corporate administrators to refuse all mail from certain geographical regions or domains because of high volumes of spam, viruses, phishing mail, and so on. They may even reject email from any source not currently known to them, though in that case there's usually a mechanism by which outsiders can apply to be included on the corporate whitelist. They may add offending addresses to an in-house blacklist, or they may use one of the many open DNS blacklists. All these approaches have their downsides, in that they can entail a risk of losing mail traffic which may impact on business processes.

Institutions and individuals, however, may be able to afford these negatives (unless, of course, they are simply unaware of them). An enterprise can take decisions to block huge swathes of IP block, or all unsolicited mail, or all mail that doesn't come from a whitelisted source, on the basis of an informed risk analysis process. They may decide that the risk that they'll lose customer enquiries, mail relating to ongoing transactions, enquiries relating to freedom of information or data protection and so on, is acceptable given the nature of their spam problem.

However, that raises a question: is it acceptable for an ISP – or, indirectly, a third party such as a Domain Blacklist (DBL) maintainer – to impose that risk upon them without consultation? Is it acceptable to do so in the case of an individual rather than an enterprise?

OPEN AND SHUT CASES

Open blacklists are a mixed bag: they range from highly professional services, scrupulously maintained, to zealots who are happy enough to blacklist the whole of Germany in the hope of applying enough pressure to persuade a recalcitrant domain to conform to their particular ideal.

Where a blacklist is maintained responsively and responsibly, so that the risk of false positives is minimized by prompt weeding, a reasonable balance is maintained. That balance lies between the need to apply sanctions against those who (through malice, greed or simple ignorance) are abusing Internet mail mechanisms, and a wish to spare those whose only crime is to share a gateway or other IP space with an abuser, from being punished for the sins of others.

Many blacklists are maintained by volunteers with a sincere conviction that everyone will benefit in the long term from the punishment of transgressors. If they think about it at all, they may assume that incidental damage to innocent parties is a regrettable, but acceptable risk. They may even count consciously on such parties to apply referred upward pressure to the service providers and administrators who cling to discredited practices. Certainly, many of us will sympathise with the discouragement of unsecured relays, indiscriminate distribution of misdirected virus notifications, poor Non-Delivery Report practice, and so on.

Essentially, most blacklists prioritise what are perceived as the interests of the majority over the convenience of the individual. This Utilitarian philosophy of the greatest good for the greatest number is in some senses laudable – indeed, you could argue that the societies that most of us live in are to some degree founded on it – but it sits uneasily in the commercial context of a mutually agreed contract.

It's probably no coincidence that those lists that come closest to striking an acceptable balance between the blocking of bad mail and the free passage of good mail are those that have at least one foot in the commercial spam-management sector. Volunteer blacklists are not always updated promptly, or responsive to pleas from those who suffer collateral damage that they aren't responsible for someone else's mismanagement. They can usually fall back on the argument that 'We don't block anyone, we just publish a list'. Those that supply a contracted service, however, have a commercial interest in maintaining good and responsible relations with customers *and* potential customers, and certainly can't afford to keep failing to meet service level agreements.

INDUSTRY PRACTICE OR BEST PRACTICE?

It's not unknown for ISPs to risk blocking some legitimate mail, in the hope of reducing spam received by their customers to zero, or as near to zero as possible. Verizon, for example, has been quoted as saying that it 'block[s] narrowly' using 'methods that are consistent with industry practices'.

But is it *best* practice? Many organizations cannot accept the risk of life- or business-critical services being disrupted by false positives, and the better anti-spam services generally try to accommodate that need by taking strenuous measures to try to ensure that no legitimate mail is lost.

Specialist reputation services try to avoid blocking mail from non-spammers who happen to share IP space with spammers, and they apply fallback mechanisms such as quarantining suspect mail. This not only allows the customer some means of monitoring performance, but lets them retrieve mail that has been incorrectly classified as spam.

Home users don't usually do that sort of risk assessment, and often have unrealistic expectations that anti-spam services will not only block all their spam, but also give them access to all their legitimate mail – two expectations that are not necessarily compatible.

However, the reported reactions [3] of *Verizon* customers to this class action and the debate around it suggest that some home users need and expect reliable mail delivery. Some of them use their connection for business, and need reliable mail delivery just as much as *Fortune 100* companies do. Furthermore, it seems that even some recreational users, if forced to think about it, prefer to receive some spam rather than risk not receiving mail from family or friends.

CONCLUSION

Home and small business users are unlikely to demand the same guaranteed levels of service delivery that are built into major corporate contracts. In fact, they just want the provider to take care of it so that they don't have to think about it. They may not bother to read their provider's Terms of Service. But they do consider their ISP accountable for the safe delivery of legitimate mail, even for a basic service.

Smaller customers are starting to realize that they may need more flexibility (even if they have to pay extra for guaranteed delivery) and to know more about how the service they're receiving works.

If ISPs want to maintain a one-size-fits-all spam-filtering service, they need, as a minimum, to make clear what users can expect of the basic service, what optional extras are available, and what your money gets you in each case. Expectation management is key: it's no longer enough to say 'we block spam; how we do it isn't important'. In order to avoid legal action and maintain market share, ISPs need to realize that spam management is a balancing act. It's also an exercise in PR.

If ISPs really want to maintain a draconian level of spam filtering, they may want to consider ensuring that they whitelist organs like *The Register* with a reputation for voicing the concerns of the end-user.

REFERENCES

[1] http://www.pcworld.com/resource/article/0,aid,119717,00.asp.

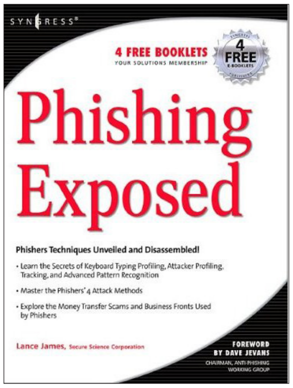
[2] http://www.spamhelp.co.uk/2005/01/verizon-spam-filtering-statement.html.

[3] http://www.theregister.co.uk/2005/01/21/verizon_class_action/.

BOOK REVIEW

PHISH FINGERING

David Harley
Small Blue-Green World, UK



Title: Phishing Exposed
Author: Lance James
Publisher: Syngress
ISBN: 1-59749-030-X
Cover Price: $49.95

There are several things that might put you off about this book. The back cover blurb ('Uncover secrets from the dark side'), the emphasis on attack code and a whiff of breathless, l33tspeak about some of the prose all tend to grate on the sensibilities of an ageing AV researcher.

However, in this case it is worth looking past the abundance of exclamation marks. *Syngress* seems to prefer to publish books by IT professionals with hands-on expertise, rather than by career authors and journalists. Lance James is a prolific contributor to anti-phishing forums such as the Anti-Phishing Working Group, he represents a security software company that is active in the tracking of phishing groups, and he is certainly a hands-on kind of guy.

TARGET AUDIENCE

You shouldn't judge a book by its cover. But the cover may be the only selection criterion available to a prospective buyer. In this case the cover tells us that the reader can, among other things, uncover phishing servers, blind drops and CSS attacks, learn how email addresses are harvested, exploit SSL and 'untangle the intricate web of international money laundering'.

The choice of marketing hooks, the blurb and the foreword all seem to indicate that the book is aimed primarily at those concerned with phishing management at a technical level – particularly programmers, law enforcement professionals, and the security community. So I came to this review with two questions: 'Does the book work for its target audience?', and 'Is anyone else likely to benefit from it?'

STRUCTURE

Following brief biographies (of the author, technical reviewer and foreword contributor), the author's acknowledgements and contents, the book is introduced by a short and to-the-point foreword by the estimable Joe Stewart, another well-known name in phisher and botnet hunting circles.

Phishing Exposed uses a characteristic *Syngress* chapter format. Each chapter includes a short introduction. Main sections are referred to quaintly as 'solutions', even when they actually describe exploits, and are interspersed with boxed 'tricks of the trade', 'tools and traps' and 'notes from the underground'. There are also copious figures (mostly screenshots).

Each chapter ends with a summary, a 'solutions fast track' consisting of three or four main points, and an oddly named FAQ section. The questions are intended primarily to test comprehension of the preceding material, so probably aren't really asked frequently. However, they do serve as a useful summary of core concepts, and the reader can request answers to specific questions from the author by submitting a form on the *Syngress* website.

Chapter 1 ('Banking on phishing') covers spam classification, cyber-crime evolution, a definition of phishing, and finishes with a section on fraud, forensics and the law.

The section on spam classification doesn't, as you might expect, involve a detailed taxonomy: rather, it starts with a brief note on identifying spammers and gangs, and goes on to say that there are eight top-level spam classifications. Oddly, it lists only four of the top-level spam classifications: unsolicited and non-responsive commercial email (UCE, NCE), list makers and scams. If you already know what a hashbuster is, you probably won't learn much from this, and even neophytes won't learn all they need to know. While we are told that 419s and auction fraud are not phishing, we are not provided with an explanation as to why they are not included in this category.

The content of the next few pages is a generalist look at phishing with some statistical content. A boxout and table compare phishing emails and phishing malware – a term used here to refer only to keyloggers. The legal section is focused entirely on the USA, though some of the more general discussion is relevant to all jurisdictions.

MORE PHISH TO PHRY

The author seems more comfortable with Chapter 2 ('Go phish!'), which focuses on three types of attack: impersonation, forwarding and popups. Readers of *VB* may be less comfortable with the level of detail provided in the attack descriptions and code, though only the most clueless of script kiddies will find much of this information new. End users who get through this section, on the other hand, will benefit in terms of an understanding of basic phishing mechanisms.

Chapter 3 ('E-mail: the weapon of mass delivery') is divided into sections on email basics, anonymous email, address harvesting, and sending spam. Much of the content is quite general. This could be a useful introduction to spam technology, discussing such issues as header forgery, open relays and proxies, though the general reader's eyes might glaze slightly at the liberal (and largely unexplained) use of regular expressions in command lines. The same reader might, however, benefit from the short descriptions of some spammers' tools and of *Spam Assassin* that follow. SPIM is mentioned in the Fast Track, but not considered in depth.

POACHERS VS GAMEKEEPERS

Chapter 4 ('Crossing the phishing line') starts with a fairly high-level description of the web, dealing with DHTML and HTTP, including a brief note on request methods, one of those topics 'everyone knows about' and no-one ever explains.

The section on misplaced trust looks at the issue of 'consumer misdirection', or the ways in which the marketing departments of banks and other organizations make the phishers' jobs easier by continuing to use long, complex links, 'click here' links, misadvertised links, arbitrary redirects and unpersonalized message text. This section (or at least a summary with less jargon and exploit code) should be used to wrap the sandwiches of many a financial marketroid. CSS attacks are alluded to, but not considered in depth.

On the *SecureScience* website (www.securescience.com), the book is described as a 'view from both sides of the phishing playing field'. It is in Chapter 5 ('The dark side of the web') that the book comes nearest to meeting that description. This chapter considers Dynamic HTML and DOM in depth, and includes information on URL poisoning, filter evasion, SSL misuse, frame attacks and session hijacking.

YOU MAY GROW UP TO BE A MULE

Chapter 6 ('Malware, money movers, and ma bell mayhem!') starts with a good section on mule recruitment and money laundering. Given the very variable quality of available information on these issues, any general reader might benefit from the information here. The size of the mule recruitment problem is largely underestimated and often goes unmentioned in security books and on informational websites.

James then goes on to consider telephony issues such as Caller ID spoofing and anonymous VoIP, mostly in the context of mule driving. The section on malware is a reasonably accurate introduction to the subject (at least as far as the past two to three years are concerned), and makes some valid points about changing patterns in malware technology and the consequent difficulties for anti-virus technology.

For Chapter 7, the author was unable to resist the title 'So long, and thanks for all the phish!'. The chapter includes some more US-centric legal observations, a fairly superficial survey of anti-phishing vendors, and some statistical observations.

DOES THE BOOK KEEP ITS PROMISES?

While the book is more detailed (and accurate) than the average *Dummies Guide*, it doesn't really constitute a complete toolkit either for the skiddie or for the anti-phishing professional. Newbies will come away with more idea of how it all works and what it all means than they had before, but won't be fully equipped for a forensic career.

Phishing Exposed is a competent, largely accurate introduction to some of the more technical aspects of phishing. Lance James writes clearly, and has a good reputation in the anti-phishing circles. There are some editing and proofing anomalies which should really have been picked up during the editing process. The industry professionals working directly in this area will not learn a great deal, although non-specialists working in other areas of security may get more out of it.

Businesses targeted by phishing sites will want to look at this book. Phish-management professionals will certainly want a copy, if only to see what people further down the food chain might be reading. Most end users will probably be too intimidated by the technical detail to consider buying it, which is a pity. There is a lot of detail, though most of it takes the form of exhaustive code and historical data rather than copious explanations. However, a general user who is prepared to sift for nuggets could learn a great deal about Internet safety, fraud and email abuse.

The book would benefit from establishing clearer differentiation between old and new threats (the same could be said of many security books). It would also benefit from a glossary and a references/further reading section.

In fact, this could have been one of two very different books: a much more detailed book on the mechanics of phishing and anti-phishing technologies for aspiring specialists, or a short book for the non-specialist, shorn of some of the less useful detail, including some fairly dated attack code. The book that we actually have is not as useful as it might be to either group. Nonetheless, for now it is certainly the best book on the subject to have come my way.

CONFERENCE REPORT

EU SPAM SYMPOSIUM 2006

Sorin Mustaca
Avira, Germany

Last month the first EU Spam Symposium took place at the University of Maastricht in the Netherlands. Since this was the inaugural symposium, delegates had not set their expectations too high, but in the event the organizers did an admirable job. This forum invited a variety of technological, jurisdictional and commercial speakers from all over the world. It was also the first public event to have an ex-spammer on the programme.

Of course, like all conferences, it had good parts and not so good parts. But, overall, it was good to see that spam is no longer ignored worldwide. The only thing I was disappointed with was that the symposium did not address phishing at all – a subject that, in my opinion, should have been covered extensively.

The symposium was opened by the head of the anti-spam department of the Dutch OPTA, Danyel Molenaar, who presented 'Enforcing anti-spam law in .NL'. OPTA is a government body that regulates compliance with legislation and regulations in the areas of post and electronic communications. Danyel's message was pretty clear and I have to agree with him: 'It is effective to enforce! But cooperation is vital'.

Next on the agenda was John Graham-Cumming, who is well known for his papers and research in email filtering and (especially) spammers' tricks [*as well as in the pages of VB - Ed*]. His paper was entitled 'Three years of spam mutation' and was very entertaining. The presentation was an up-to-date overview of the latest spamming trends. All the techniques presented are also described in John's *Spammers' Compendium*, which is available on his website (http://www.jgc.org/).

Cristina Bueti from the ITU (International Telecommunication Union) spoke about how a United Nations (UN) specialized agency helps the world to communicate. Her presentation was called 'Countering spam in a digital world'.

It was very encouraging to see that organizations like the UN are taking the spam problem seriously. However, I have some doubts that anything will really happen without someone employed to enforce it. Currently, the UN is providing advice to its members on how to fight spam and other digital threats. The other good thing about this presentation was that it was the only one where I saw mention of the word 'phishing' – which was listed among the various aspects of spam.

Next, Ann Elisabeth presented her paper on tracking spammers in Norway. Unfortunately, the presentation ran over the allotted time and had to be interrupted due to time constraints. She attempted to explain which email header fields are important in tracking a spammer and how this information is used. Tools to analyse IP addresses, MX information, DNS records and many others are part of her 'bag'. Regrettably, because the presentation was cut short, the audience was left with incomplete information.

The most controversial presentation was that of 'Spammer-X', a.k.a. Eddy. In his presentation 'Inside the world of spam: from the eyes of a spammer', Eddy explained to delegates that, for five years, he was the man in the shadow who sent 'many, many millions of spam emails'. He even wrote a book called *Inside the Spam Cartel*.

I have doubts that the guy was as good as he says he was. He wrote a book about the spamming industry, so why didn't he receive questions from his old 'friends' from the cartel, from police or other law enforcement organizations? Of course, he didn't reveal any real names in his book.

Eddy managed to break the circle and to get away from 'the spam cartel', and he also kept all the money that he made. The central point of his presentation was to explain why the spamming business works: it works because people buy the products advertised in spam. Unfortunately, they don't even get the desired effects from the blue pills they buy. Why? Because, according to Eddy, all the products advertised in spams are fakes.

Jose Maria Gomez Hidalgo, of the European University of Madrid, presented 'History, techniques and evaluation of Bayesian spam filters'. I enjoyed this presentation from an academic point of view, but it was rather long and very heavy on statistics. The presentation focused on *Spam Assassin* rule sets, on Bayesian poisoning and on known methods to improve statistical filtering.

The final speaker was Matthew Prince, from anti-spam software and services company *Unspam*, who explained 'Why anti-spam laws haven't worked, and what to do to fix them'. Those who attended the *VB* conference in Dublin last year might have had a déjà vu experience. Matthew enhanced that presentation with up-to-date information and described in a very entertaining and enthusiastic way what Project Honeypot does and how it helps with tracking spammers. I have to say that he also convinced me to get the honeypot they are working on and give it a try. However, I do have some doubts about who is using this honeypot – if I were a spammer, I would install the honeypot and develop some kind of obfuscation methods to bypass it.

Slides and webcasts for all the presentations are available at http://www.spamsymposium.eu/archivewebcast.htm.