SEPTEMBER 2006

virus BULLETIN

The International Publication on Computer Virus Prevention, Recognition and Removal

CONTENTS

2 **COMMENT**
Respecting the testing

3 **NEWS**
Nearly VB 100%
Testing patience

3 **VIRUS PREVALENCE TABLE**

4 **VIRUS ANALYSIS**
Gatt got your tongue?

6 **ROOTKIT ANALYSIS**
Raising the bar: Rustock and advances in rootkits

10 **FEATURE**
The world of botnets

BOOK REVIEWS
13 War of the words
14 I spy

16 **PRODUCT REVIEW**
Kaspersky Internet Security 6.0

20 **END NOTES & NEWS**

IN THIS ISSUE

TARGET IDA

The Interactive Disassembler (IDA) is one of the most popular tools for anti-virus researchers. Now, its IDC scripting language has become the latest target for virus writers. Peter Ferrie has the details of W32/Gatt.

page 4

NEVER-ENDING STORY

The never-ending game of hide-and-seek between the anti-virus industry and rootkits has begun a new chapter. Backdoor.Rustock.A is an advanced rootkit that could be considered the first-born of a next generation of stealth malware. Elia Florio and Prashant Pathak reveal the details.

page 6

READING MATTER

David Harley gives his opinion on whether Robert Slade's *Dictionary of Information Security* and Piccard *et al.*'s *Combating Spyware in the Enterprise* are worthy of a place on your bookshelf.

pages 13 & 14

vbSpam supplement

This month: anti-spam news and events; and Sorin Mustaca looks at present and future phishing techniques.

ISSN 1749-7027

'Competition for good test results, and so for respect, trust and strong sales, feeds development and innovation.'

John Hawes, Virus Bulletin

RESPECTING THE TESTING

Last month, *ConsumerReports.org* (*CR*), the online arm of the US Consumers' Union society, announced proudly to the world its decision to create 5,500 virus 'variants', as part of an extensive test of anti-virus products. No details were provided as to what these were variants of, how they were created or verified, or how they were put to use in the tests. All that is known is that the malware was provided by *ISE* (*Independent Security Evaluators*), whose president Avi Rubin disassociated himself from the tests.

The initial reaction within the AV industry was a slow, sad shaking of heads, and raised eyebrows of disbelief. Sensible voices pointed out the flawed methodology of the tests and the availability of similar test results from specialized and respected test centres running retrospective testing. These organizations focus on the same issues as those raised by *CR*, but they use existing, real-world viruses. Pessimistic members of the AV community feared the escape of the new variants into the wild, and pondered the legal implications for their creators should they cause any damage. Analysts complained at the volume of extra work that looms, once the fat chunk of new malware finally reaches their desks for inspection and identity creation (one of the few statements issued by *CR* has suggested that handing the malware over to industry experts would be 'a good idea'). Minds were cast back to similar scandals of the past, to the tests carried out by *CNET* using the Rosenthal 'simulated' viruses, and to the infamous University of Calgary 'virus-writing' course.

Since then, the issue has mushroomed into a fizzing cloud of counter-accusations. Backlash against the virus creators' critics has mainly taken the form of accusations against the AV industry of hiding the reliance of AV products on signature-based detection and of hyping the efficacy of heuristic methods (of course, the old 'they-write-all-the-viruses-themselves' chestnut has cropped up here and there too). 'They're telling you they have all this heuristic capability, but the best they can do is 50 per cent. That's nothing; that's terrible.' So cried Peter Firstbrook, head of the cyber security division of marketing palmist *Gartner*, fresh from compiling the highly lucrative 'magic quadrant' report on the AV industry. User faith in virus protection is being battered by a hail of abuse.

Testing is important. Competition for good test results, and so for respect, trust and strong sales, feeds development and innovation. Flashy logos and catchy slogans may capture the eye and the ear, but without the credibility given by proven effectiveness, no product can hope to thrive. Dissemination of test results also helps users, allowing them to judge the performance of their product, and to demand better where it is available. To achieve these goals, testing must be credible, it must be transparent, verifiable and accountable.

VB, along with several other specialized and recognized testing organizations, provides a vital service, both to those within the industry and to their customers, ensuring that security software performs as well as it can. To provide these services the testing organizations rely on the community they serve, and abide by its ideologies and beliefs. One of the most strongly held convictions throughout the industry is that creating viruses is *never* justified. Those who do so are forever beyond the pale, barred from employment, as they are from respect and trust, by their fellows. In conniving in the creation of viruses, *CR* has damaged its credibility as surely as if it had been involved in any other criminal activity.

Here at *VB* we hope, in the near future, to improve and expand our own testing, to include an ever-growing variety of threats; with spyware on the horizon, the shadowy territory of rootkits and the legal minefield of adware lie ahead. We would like, at some point, to be able to include an element of retrospective testing into our service. In order to do all this successfully and properly, we rely on the trust and respect of our readers and of the industry we study and report on, and we will certainly not be hiring any virus writers.

NEWS

NEARLY VB 100%

In the recent *Windows XP* comparative review (see *VB*, June 2006, p.11), *VB* reported that *VirusBuster* failed to achieve the results required for a VB 100% award. After discussion with the developers, it was discovered that out-of-date virus identities had been provided with the product and that *VB*'s tester had failed to acknowledge the warning messages suggesting that an update of the virus database was in order. While the results remain valid for the version of the product tested, *VB* has since tested the product using what would have been the most recent identities for the submission deadline of the test. The two ItW viruses that prevented the product from achieving a VB 100% at the time were caught – meaning that if the correct data had been supplied originally, *VirusBuster* would easily have achieved a VB 100%.

Moving on to the latest *NetWare* comparative (see *VB*, August 2006, p.15), an unfortunate series of miscommunications resulted in *Symantec*'s product missing the submission deadline. The product has since been run against the test sets and detected 100% of samples in the ItW test set without alerting on any false positives – had the product arrived in time to be included in the comparative review, it too would easily have achieved a VB 100%.

TESTING PATIENCE

After having come in for a great deal of criticism in recent weeks over its AV testing methodology (see p.2), *Consumer Reports* has – amazingly – damaged its credibility further after having confirmed that, during its testing of anti-spyware applications, *CR* did not test against any spyware.

CR's review of anti-spyware products was based on running the applications against the *Spycar* set of applications that mimic spyware behaviour. While it would be understandable (if not entirely forgivable) that testers lacking experience in the anti-malware field could make such a gaffe, what makes this more astounding is that the creators of *Spycar* state clearly and specifically that *Spycar* is not suitable (as a sole test method) for anti-spyware testing. The EULA states '...Spycar ... is intended to be used to see how anti-spyware tools cope with new spyware for which they didn't have a signature. It is not intended to provide perfect anti-spyware tests, or to act as a substitute for any other form of evaluation.' And the *Spycar* website (www.spycar.org) reads: 'Is Spycar a comprehensive test of anti-spyware tools? No ... Spycar does not evaluate the signature base, the user interface, and other vital aspects of an anti-spyware tool. Thus, Spycar alone cannot be used to determine how good or bad an anti-spyware product is.'

After such a controversial performance *CR* will need to work hard if it is to recoup its credibility in the anti-malware field.

Prevalence Table – July 2006

Virus	Type	Incidents	Reports
Win32/Netsky	File	47,552	46.78%
Win32/Mytob	File	18,589	18.29%
Win32/Bagle	File	17,659	17.37%
Win32/Mydoom	File	8,180	8.05%
Win32/MyWife	File	5,124	5.04%
Win32/Bugbear	File	1,500	1.48%
Win32/Pate	File	1,487	1.46%
Win32/Lovgate	File	303	0.30%
Win32/Sdbot	File	172	0.17%
Win32/Zafi	File	169	0.17%
Win32/Mimail	File	119	0.12%
Psyme	Script	104	0.10%
Win32/Sality	File	91	0.09%
Win32/Feebs	File	90	0.09%
Win32/Dumaru	File	87	0.09%
Win32/Gibe	File	75	0.07%
Win32/Valla	File	67	0.07%
Win32/Maslan	File	66	0.06%
Win32/Mabutu	File	60	0.06%
Win32/Klez	File	33	0.03%
Win32/Reatle	File	25	0.02%
Wonka	Script	21	0.02%
Win32/Swen	File	10	0.01%
Win32/Gael	File	9	0.01%
Win32/Kriz	File	8	0.01%
Win32/Areses	File	7	0.01%
Win32/Chir	File	7	0.01%
Win32/Elkern	File	6	0.01%
Win32/Small	File	6	0.01%
Yamanner	Script	5	0.00%
Thus	Macro	4	0.00%
Win32/Nimda	File	4	0.00%
Others[1]		22	0.03%
Total		101,661	100%

[1]The Prevalence Table includes a total of 22 reports across 18 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

VIRUS ANALYSIS

GATT GOT YOUR TONGUE?

Peter Ferrie
Symantec Security Response, USA

As operating systems have become more secure (or at least less insecure), virus writers have started to attack applications instead. One of the most popular tools for an anti-virus researcher is the *Interactive Disassembler* (*IDA*), and its IDC scripting language has become the latest target, thanks to W32/Gatt.

THE IDC LANGUAGE

.IDC files are script files that can control *IDA* by using the IDC scripting language. The IDC language is very C-like in appearance, and supports functions, variables, etc. – all of the things that one would expect from a good scripting language. However, as with *Microsoft*'s VBScript and JScript scripting languages, IDC files are compiled at the moment they are requested to run, and the resulting binary form is executed directly in memory. There is even a built-in Compile function, to perform dynamic compilation of IDC files.

GATTMAN AND BOBBIN

W32/Gatt is a polymorphic entry-point obscuring infector of these IDC files. It begins by allocating a one-megabyte(!) buffer for the new decoder. That might sound like overkill, but in fact the generated decoders often require more than half of that buffer. However, there is nothing in the generator to prevent a decoder from exceeding the buffer. If that were to happen, the virus would simply crash, since it contains no exception handling code.

After the allocation, the virus attempts to create a file mapping of itself, and here is the first bug: even if the mapping operation fails, the virus still attempts to infect files. Another, similar bug follows immediately: even if the attempt to map a view of the file fails, the virus still attempts to infect files. Additionally, if any handle cannot be closed for any reason, the allocated block is never freed explicitly.

WARP FACTOR NINE

Assuming that all goes well, the virus will generate a new decoder. Despite appearances, the decoder is only lightly polymorphic. The polymorphic engine is capable of producing random comments of both the '/**/' and '//' style, including comments that span multiple lines. For the first comment style, which is designed to support multiple lines already, no special handling is required. For the second comment style, which is intended to be only a single line, the virus ends the line with a backslash line-continuation character.

Each of the tokens can also be split randomly across lines, by using the backslash line-continuation character. In an extreme case, it would be possible for the virus to produce files where only a single character appears on each line, but this is unlikely to occur. The '/**/'-style comments can also appear between the tokens. Finally, non-token elements – variables, and string elements – have their case mapped randomly.

This is essentially all that the polymorphic engine does. The only other variation is in the way in which the virus chooses to rebuild itself.

TESTING, ONE, TWO... OOPS

Not surprisingly, the polymorphic engine is full of bugs. The decoder begins with a conditional expression, which tests whether a variable that the virus declares has a value of 0. The virus carries seven variations of this expression: two 'if' forms, two 'while' forms, and three 'for' forms. The bug occurs when selecting the form to use: the engine uses the 'test' instruction instead of the 'cmp' instruction.

This bit-wise comparison results in two variations of the expression that cannot be selected. One of those unselectable blocks contains a bug anyway: a missing semicolon character means that the line would generate a syntax error during compilation, and the execution will not occur. As if that wasn't bad enough, one of the remaining selectable conditional expressions also contains a bug. That bug is also related to a semicolon character. However, this time the bug is not that the semicolon character is missing, but that it is in the wrong place. Again, the line would generate a syntax error during compilation, and the execution will not occur.

THE WRITE WAY

The virus works by converting IDC files into droppers of a *Windows* executable file. This executable file is what performs the infection of other files – IDC files infected with W32/Gatt do directly infect other IDC files. To try to hide the executable file within the IDC file, the virus encodes the executable file into randomly sized blocks, and writes them out individually. This is as opposed to some viruses for other file formats, which declare an array of some kind to hold the entire file as a single block.

In the case of W32/Gatt, eight-bit values can be written by using 'fputc', followed by the literal character, or encoded

in '0x' form. 16-bit values can be written using the 'writeshort' function, and 32-bit values can be encoded using the 'writelong' function. These last two functions only accept the value in '0x' form. These functions also accept a parameter that describes the endianness of the value. The virus selects the endianness randomly, and encodes the value in the appropriate order.

Otherwise, values can be written using the 'writestr' function. Another bug exists here: if the 'writestr' function is used to write the final character in the file, the engine will crash.

THE SEARCH BEGINS

Once the new decoder has been generated, the virus begins the search for files to infect. The file enumeration is done by using the usual recursive subdirectory searcher. The virus wants to find any file whose suffix is '.IDC'. The difference here is that the suffix is not compared directly. Instead, the virus uses the SHA-1 algorithm to create a hash of the suffix, and compares that hash to one that the virus carries. This might have slowed down analysis a little bit, to determine the file type of interest, if the virus author hadn't made it quite clear what kind of file the virus wanted to infect.

The virus has no infection marker. The nearest thing to an infection marker is a check of the size of file that has been found. Any file larger than 419,430 bytes (0x66666 in hex) is considered to be infected. If a file is not infected already, then the virus searches within it for the string 'static', which the virus assumes is the start of a subroutine. If that string is found, then the virus examines the text between the first left and last right brace characters that it sees in that subroutine, counting all of the semicolon characters that it sees.

The virus also watches for the 'for' token, since it also contains semicolon characters, but they must not be counted. The virus recognizes the last right brace by incrementing a brace count for each left brace that is seen after the first one, and decrementing the count for each right brace that is seen. Once the count reaches zero, the virus stops looking.

Once the last right brace is seen, the virus chooses randomly from the count of semicolon characters, and inserts the virus code after the nth semicolon, which makes the virus entry-point obscuring. A critical bug occurs here: if any file is infected, the stack is unbalanced because of some leftover code. Specifically, the parameters for a particular API have been pushed onto the stack, but presumably during 'optimization' of the code, the API call was moved into a subroutine. This subroutine pushes the parameters locally, so the old parameters remain on the stack. Because of this bug, the virus crashes immediately after a single infection. Perhaps the virus author tested only on a single file at a time, and so never noticed the problem.

MAKING A HASH OF THINGS

If the current file is not an .IDC file, then the virus hashes the full filename and compares it to a list of five hashes that the virus carries. The reason for this check was clear even before the hashes were decoded: recognizable packer switches are present in the virus body, and though they are never used, it gave me a clue about the probable filenames. Three of the hashes were easy to guess, and they correspond to three runtime compressors (EXE32PACK.EXE, PEPACK.EXE, UPX.EXE). The other two yielded very quickly after a brute-force attack. One is a file manipulation tool called VGALIGN.EXE, but the other is an unknown tool called SPEC.EXE.

If one of these files is found, then the virus attempts to copy itself into the directory that contains that file. A bug exists here: the copy will fail if both files are present in the same directory, and in that case, the virus will keep searching for files.

If the copy succeeds, then the virus executes the file that corresponds to the hash, passing the virus filename as a parameter. The idea here is to use one of these tools to change the appearance of the file, and then to regenerate the decoder using the new file. However, yet another bug exists here: process execution is asynchronous, but the virus does not wait for the new process to complete before attempting to access the virus file again. Thus, the original virus file is used to generate the decoder, resulting in all encoded files having the same appearance.

CONCLUSION

On the day that the virus author released the virus, he posted a message on his website that said the virus 'will be very hard for AVers to detect'. Later that same day, we started detecting it. The following day, the virus author changed the message to one that said the virus 'will not be released'. I'd like to think that it's not a coincidence – it might look polymorphic to him, but it doesn't to me.

W32.Gatt

Type:	Polymorphic entry-point obscuring file infector.
Size:	16,384 bytes (EXE), varies (IDC).

ROOTKIT ANALYSIS

RAISING THE BAR: RUSTOCK AND ADVANCES IN ROOTKITS

Elia Florio
Symantec Security Response, Ireland

Prashant Pathak
Symantec Security Response, USA

The never-ending game of hide-and-seek between the anti-virus industry and rootkits has begun a new chapter. Recently a new type of rootkit was discovered in the wild and it is unique given the techniques it uses.

Backdoor.Rustock.A is an advanced rootkit that could be considered the first-born of a next generation of stealth malware, thanks to the special characteristics it possesses. It uses a mixture of old techniques and new ideas that, when combined, make a piece of malware that is stealthy enough to remain undetected by many commonly used rootkit detectors (such as *RootkitRevealer*, *BlackLight*, *IceSword*, *DarkSpy* and *GMER*).

We can consider this rootkit to be an advanced example of 'stealth-by-design' malicious code [1]. At the time of writing this article, it has been reported that a new variant of this malware (Rustock.B [2]) has been discovered in the wild and, although it is similar to its predecessor in certain aspects, this rootkit is significantly improved. However, this article will focus mainly on the .A variant.

WHY IS RUSTOCK.A SPECIAL?

Table 1 summarizes the reasons why Rustock.A is considered to be an advanced rootkit. It shows in the left-hand column all the typical anomalies detected by rootkit detectors and in the right-hand column the countermeasures adopted by Rustock.A to avoid detection.

The MSR_SYSENTER hook technique is not new, it has already been documented in Greg Hoglund's book *Subverting the Windows kernel* [3], but Rustock.A is the first piece of malware using this method to have been discovered in the wild.

In addition, the use of the Alternate Data Stream (ADS) as a storage area for the malicious driver gives the rootkit several advantages. In fact, by exploiting the ADS, the rootkit does not have to worry about file hiding, because the SYS file is hidden by the *Windows* operating system. The rootkit just prevents access to the ADS by locking it with system privileges and by hooking some special NTFS IRP functions that control create/delete operations on this stream. Manual removal of this threat is not a trivial task, because the rootkit works in safe mode and booting from a recovery console does not allow users to manipulate the ADS.

Rootkit detection	Rustock countermeasure
Detection of hidden processes	Rustock.A has no process; the malicious code runs inside the driver and in kernel threads.
Detection of hidden files	Rustock.A does not hide files; it uses the NTFS Alternate Data Stream to store its driver. In addition, the rootkit prevents access to the ADS by locking it.
Detection of registry keys	The rootkit controls ZwSaveKey and intercepts any program that tries to dump the registry to a file. It also can add/delete its registry subkey based on a specific event (e.g. IOCTL code detection during DeviceIoControl).
Detection of hidden driver	The rootkit removes its entries from many modules' kernel structures including the Services Control Manager, Object Manager, and the loaded module list so that this enumeration fails.
Detection of Native API hooks	Rustock.A does not hook or patch system calls directly, it gains control by hooking the MSR_SYSENTER routine and other IRP functions.
Detection of SDT hooks/changes	The rootkit does not alter Service Descriptor Table pointers on a global basis, but rather modifies them on a per-thread basis.
Detection of MSR_SYSENTER hook	Rustock.A modifies the address of MSR_SYSENTER and also patches a routine of the *Windows* kernel where MSR_SYSENTER is loaded and checked.
Detection of the malicious file	The SYS driver uses a polymorphic packer to scramble its code and appears different from sample to sample.

Table 1: Reasons why Rustock.A is considered to be an advanced rootkit.

The Rustock installer comes as an executable file with a size of around 65–70 KB and is scrambled by a polymorphic packer which mixes NOP-equivalent and floating-point opcodes together with real instructions. The executable drops the %TEMP%\pe386.sys file and then uses the CreateService and StartService APIs to run it as a service. This SYS module is compiled as a kernel-mode DLL [4] and actually works as a loader. It decrypts and decompresses the real rootkit driver inside a buffer allocated in kernel memory by using ExAllocatePool. The malware

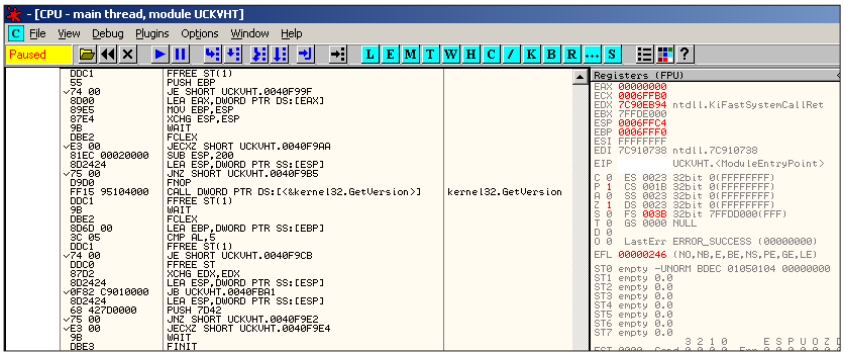


Figure 1: Example of Rustock.A scrambled code: useless instructions are marked in red.

copies itself inside an ADS storage that should have the random generated name '\System32:[RND_NUMBERS]', but due to a bug (probably the lack of initialization of the random numbers generator) the ADS is always named '\System32:18467' on *Windows XP*.

SYSENTER HOOKING

On *NT/2k* systems control is transferred from user to kernel mode via software interrupt INT 2E. On *Intel/AMD* platforms, which support the SYSENTER/SYSCALL instruction, *XP* and above systems use SYSENTER/SYSCALL to transfer control from user to kernel mode. Rustock uses both SYSENTER and IDT hooks to execute code every time a system call is made. The modified SYSENTER/IDT handler hooks every thread that attempts to execute any of the below-mentioned system calls. Thus, the rootkit hijacks the system calls on a thread-level basis

```
.text:        New2E0ffsetAddress:                     ; DATA XREF: InstallSysEnterAndIDTHooks+
.text:                        pusha
.text:                        pushf
.text:                        push    fs
.text:                        push    ds
.text:                        mov     bx, ss:Kfsvalue
.text:                        db      66h
.text:                        mov     fs, bx
.text:                        mov     bx, ss:Kdsvalue
.text:                        db      66h
.text:                        mov     ds, bx
.text:                        call    HookSSDT
.text:                        pop     ds
.text:                        pop     fs
.text:                        popf
.text:                        popa
.text:                        jmp     Old2E0ffsetAddress
.text:        ; ---------------------------------------------------------------------------
.text:                        align 4
.text:
.text:        NewSysEnter:                            ; CODE XREF: .text:
.text:                                                ; .text:
.text:                                                ; DATA XREF: InstallSysEnterAndIDTHooks+
.text:                        pusha
.text:                        pushf
.text:                        push    fs
.text:                        push    ds
.text:                        mov     bx, ss:Kfsvalue
.text:                        db      66h
.text:                        mov     fs, bx
.text:                        mov     bx, ss:Kdsvalue
.text:                        db      66h
.text:                        mov     ds, bx
.text:                        call    HookSSDT
.text:                        pop     ds
.text:                        pop     fs
.text:                        popf
.text:                        popa
.text:                        jmp     OldSysEnter
```

Figure 2: Code snippet showing INT 2E and SYSENTER hook.

rather than using KeServiceDescriptorTable to hook on a global basis.

It hooks the following system calls to hide service-related registry keys and CPU usage:

- ZwOpenKey
- ZwEnumerateKey
- ZwQueryKey
- ZwCreateKey
- ZwQuerySystemInformation

The rootkit modifies the output of the ZwOpenKey API in a unique way. It does not modify the output if the calling process is services.exe. For all other processes, if a process is attempting to open the pe386 key, ZwOpenKey returns an error code of 'STATUS_OBJECT_NAME_NOT_FOUND'. Hence, no process other than services.exe can obtain a handle to Rustock's service keys. Rustock modifies ZwEnumerateKey by incrementing the index value by one whenever a key containing subkey pe386 is enumerated. Rustock modifies ZwQueryKey to decrement the value of the Subkeys field in the output when a key containing subkey pe386 is queried with the KeyFullInformation option.

Rustock modifies the output of ZwCreateKey in a similar way to ZwOpenKey. It does not change the output if the calling process is services.exe. For all other processes attempting to create a key named pe386, ZwCreateKey returns an error code of 'STATUS_OBJECT_NAME_NOT_FOUND'. The modified ZwQuerySystemInformation API zeros out the user and kernel mode usage time for services.exe and adds it to the first process in the list (which is the system idle process). Since Rustock injects spam-mailing code in services.exe, zeroing out user and kernel mode usage for services.exe would not raise any suspicion in a user monitoring the system performance using tools like *Process Explorer*.

INVISIBLE MODULE IN WINDOWS KERNEL

Rustock's driver is stealthy when loaded in kernel memory. The rootkit attempts to hide its presence using the techniques shown in Table 2.

IRP PATCHING TECHNIQUE

Every driver object sets IRP handlers that are invoked whenever a particular type of IRP is generated. These IRP handlers are a set of function pointers stored in per-driver data structure DRIVER_OBJECT. For example, when a

Rootkit technique	Description
Unlinking the driver loaded from the module list	Every driver in the *Windows* kernel has a per-driver data structure, namely DRIVER_OBJECT. The driver object structure has a DriverSection field that contains a doubly linked list of all loaded modules. A driver receives a pointer to its driver object in the startup routine. Rustock.A parses the linked list stored in DriverSection (DRIVER_OBJECT-> DriverSection- >InMemoryOrderLinks) and unlinks itself from the list. This causes the driver to be hidden from all user and kernel mode enumeration using this list (ex: PsLoadedModuleList API).
Deleting the driver object from Object Manager Namespace	Rustock obtains the \Driver directory object using the ObQueryObjectByName API. The \Driver object contains a list of driver objects that are loaded. Rustock parses the list and unlinks any object from the \Driver directory if the object name matches its driver name. Hence, all APIs that enumerate from this list fail to enumerate Rustock's driver.
Unlinking itself from the service list	Whenever a service is created, it is SCM registered with Service Control Manager (SCM). SCM maintains a linked list of services registered with it. When the status of a service is queried, SCM parses this linked list to find information regarding a particular service. By using DKOM techniques, Rustock deletes its service-related entry in this list. Once a service entry is removed from this list, all service enumerations in user and kernel mode fail to list the running service.

Table 2: Techniques used by Rustock.A to hide its presence.

```
.text:          RemoveFromPsLoadedModuleList proc near  ; CODE XREF: DriverEntry+
.text:
.text:          DriverObject    = dword ptr  4
.text:
.text:                          mov     eax, [esp+DriverObject]
.text:                          mov     eax, [eax+DRIVER_OBJECT.DriverSection]
.text:                          mov     ecx, [eax]
.text:                          mov     eax, [eax+4]
.text:                          mov     [eax], ecx
.text:                          mov     [ecx+4], eax
.text:                          retn    4
.text:          RemoveFromPsLoadedModuleList endp
```

Figure 3: Code snippet that unlinks the driver from the loaded module list.

```
.text:                          mov     eax, offset Ntfs_Irp_MJ_Query_Information
.text:                          lea     ecx, [esi+1Ch]  ; IRP_MJ_QUERY_INFORMATION
.text:                          xchg    eax, [ecx]
.text:                          mov     Old_Ntfs_Irp_MJ_Query_Information, eax
.text:                          jmp     short loc_1270C
.text:          ; ---------------------------------------------------------------------------
.text:
.text:          loc_126FD:                              ; CODE XREF: PatchFileSystemIrpsQueryAndCreate+
.text:                          mov     eax, offset sub_12106
.text:                          lea     ecx, [esi+68h]
.text:                          xchg    eax, [ecx]
.text:                          mov     dword_148E0, eax
.text:
.text:          loc_1270C:                              ; CODE XREF: PatchFileSystemIrpsQueryAndCreate+
.text:                          mov     eax, offset FileSystem_Irp_MJ_Create
.text:                          add     esi, 38h        ; IRP_MJ_CREATE
.text:                          xchg    eax, [esi]
.text:                          mov     ecx, [ebp+FileObject] ; Object
.text:                          mov     Old_FileSystem_Irp_MJ_Create, eax
.text:                          call    ds:ObfDereferenceObject
```

Figure 4: Code snippet that patches NTFS driver object's IRP handlers.

create event occurs for a device, I/O Manager creates an IRP of type IRP_MJ_CREATE and invokes the corresponding handler for the driver object that is registered with the device.

Rustock.A makes use of IRP patching for two main reasons: bypassing resident firewalls at low level and protecting the Alternate Data Stream that stores the malicious driver. Once Rustock is installed on a machine it uses the machine's network connection to send spam messages. To avoid triggering any detection by firewalls present on the compromised system, Rustock patches IRP_MJ_CREATE and IRP_MJ_QUERY_INFORMATION handlers for TCP/UDP drivers.

In addition, it patches IRP_MJ_CREATE and IRP_MJ_QUERY_INFORMATION handlers for the NTFS file system driver. Hence, any application that uses the file system stack will not be able to query for the ADS file and will not be allowed to delete an ADS with the same name.

BYPASSING ROOTKIT DETECTORS THAT USE A CROSS-VIEW APPROACH

Many rootkit detectors use a cross-view-based detection algorithm. This means that they detect hidden objects by finding the discrepancies between a high-level view and a low-level view.

For example, a simple rootkit detector can enumerate the list of processes using a method similar to Windows Task Manager, and then it will try to enumerate the processes again using different low-level methods. If everything is fine, the obtained lists will not have differences or discrepancies. A similar approach can be applied also with files or registry keys enumeration.

The strength of this method is that it is totally generic and it doesn't need to know how a particular rootkit works. The cross-view detection does not care about the type of hooks used or the kernel objects altered, it just looks for discrepancies and anomalies, so it can uncover all the common rootkits easily. However, since the cross-view detection is based on a specific enumeration algorithm, when that algorithm is disclosed, any rootkit can attempt to find a workaround to bypass the detection and maintain its invisibility.

A similar situation has already happened with the 'FUTo' rootkit [5]. Having found out that *BlackLight*'s detection of hidden processes relies on OpenProcess() API, a group of researchers tried to design a modified version of the popular 'FU' rootkit, enhanced sufficiently to avoid detection by *BlackLight*.

In a similar case recently, a person with the alias 'PE386' (probably the same person who created the Rustock rootkit)

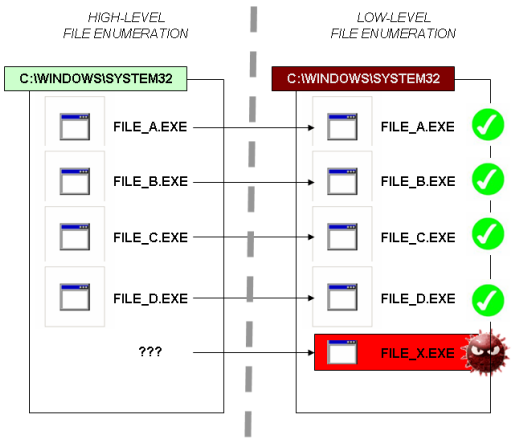


Figure 5: Generic diagram showing how the cross-view detection works.

posted on the rootkit.com website a proof-of-concept code [6] that implements a special system hook that is able to hide files from *RootkitRevealer* and *BlackLight*. The strength of these rootkits is that they are designed specifically to be stealthy and to evade the detection method.

ADVANCED ANTI-DETECTION TECHNIQUES

Rustock.A hooks the following system calls to avoid being detected by anti-rootkit programs:

- ZwSaveKey
- ZwDeviceIoControl

It modifies the behaviour of ZwSaveKey. Whenever any registry hive containing \machine\system, i.e. HKLM\System, is saved to a file, Rustock creates a new registry hive. The registry hive is loaded under HKLM\pe386, which is a copy of HKLM\System\CurrentControlSet\Services\pe386 and it deletes the original copy. A cross-view-based detection algorithm would not reveal anything, as nothing seems to be hidden. Once the routine completes, it creates a pe386 key and hides from registry enumeration API.

It also modifies the behaviour of ZwDeviceIoControl. This change is targeted specifically towards *Kernel SC*, a tool used to detect the presence of hidden services. Rustock modifies the behaviour of ZwDeviceIoControl only when the I/O control code is 0x22265A and the target driver is Kernel SC (knlsc). It creates a new registry hive under HKLM\pe386 which is an exact copy of the data contained in the IOCTL output buffer. Once the routine has completed Rustock unloads the HKLM\pe386 registry hive to remain hidden.

Finally, Rustock.A obtains a list of loaded kernel modules and searches for filtnt.sys, which is the kernel mode component of *Outpost Firewall*. It then patches the import table address of IoGetCurrentProcess to point to a new function. The modified IoGetCurrentProcess returns the EPROCESS block of the initial system process when the current process is services.exe. This modification possibly results in *Outpost Firewall* not triggering on an active connection from services.exe, but due to time constraints we are not able to confirm this.

CONCLUSIONS

All of the features that we have mentioned here make Backdoor.Rustock.A totally invisible on a compromised computer when installed. It even seems able to achieve all of its stealth functionality without problems on a beta version of *Microsoft Windows Vista* (6.0.5270).

We believe that Rustock.A and .B variants are probably Russian creatures, part of a large project of stealth spam malware called 'SpamBot'. Both the rootkits contain the strings 'G:\bot-mailer\007spambot-01\driver\objfre' and 'Z:\NewProjects\spambot\last_beta\driver\objfree', which leads us to believe that we will see new versions of this malware in the future.

REFERENCES

[1] Rutkowska, J. Rootkits vs. Stealth by design malware. http://www.invisiblethings.org/papers/rutkowska_bheurope2006.ppt.

[2] Backdoor.Rustock.B analysis. http://securityresponse.symantec.com/avcenter/venc/data/backdoor.rustock.b.html.

[3] Hoglund, G. Subverting the Windows Kernel. http://www.rootkit.com/.

[4] Roberts, T. DLLs in Kernel Mode. http://www.wd-3.com/archive/KernelDlls.htm.

[5] Silberman P, CHAOS. 'FUTo' http://uninformed.org/?v=3&a=7.

[6] PE386. Hiding files from RootkitRevealer and F-Secure BlackLight. http://www.rootkit.com/newsread_print.php?newsid=526.

[*Aleksander Czarnowski will present a paper on anti-rootkit safeguards and methods of their bypassing at this year's Virus Bulletin Conference in Montréal (11–13 October). To read the abstract, view the rest of the conference programme and book your place online see http://www.virusbtn.com/conference/vb2006/.*]

FEATURE

THE WORLD OF BOTNETS

Dr Alan Solomon, Programmer, UK
Gadi Evron, Beyond Security, Israel

With a Trojan horse on one compromised computer, you would be able to do whatever you wanted. That computer would be as good as your own. You would own it. Now imagine that you owned 100,000 such computers, scattered all over the world, each one running and being looked after in someone's home, office, or school. Imagine that with just one command, you could tell all of these computers to do whatever you wanted.

You could tell them to access the same website simultaneously, to search for security weaknesses in nearby computers, or just to send out large amounts of spam. If you sent out one spam message from each computer every second, you could send a million pieces of spam in ten seconds; hundreds of millions per hour. You would effectively be in command of an army of robots, or bots for short. The actions taken by these bots would appear to be the actions of the individuals whose computers were compromised.

Now imagine that you own a million compromised machines. You would be in possession of a force capable, at the very least, of significantly slowing down activity on the Internet or taking down a large Internet-based business. This, of course, would be illegal, but that does not stop some people from doing it. Today, there are 3.5 million bots on unique IP addresses used every day for spam purposes alone.

BOT-HERDING HOW TO

The first step towards becoming a bot-herder is to google the subject, and proceed to recruit an army – a botnet (also called a drone or zombie army). Some of the most commonly used recruitment methods include sending out millions of email messages that contain something tempting to click on, or setting up websites with browser exploits that activate a drive-by (the surfer merely has to access the web page for a trojan to be installed on their computer silently, without any interaction). Another successful method of bot recruitment is scanning networks for vulnerable computers. Not everyone installs patches, and as a bot-herder you don't care whose computer you recruit for your bot army; the name of the game is quantity, not quality.

You'll also need to put some effort into maintaining your army, as there are many rivals, obstacles and enemies on every corner. Users may run one of these pesky anti-virus products that discover and remove your bot, their computer may crash, and they may reinstall *Windows*. Alternatively, another botnet controller or bot could come along and hijack your control of the user's computer. But that's no great problem – there are a lot of potential victims out there, you just need to keep recruiting. There's no need to worry about losing the grunts when there are a lot of fresh troops out there without criminal records (such as having been blacklisted for sending spam).

Your biggest danger as a bot-herder is being discovered, unveiled and arrested. This has happened to other botnet controllers, but it's a rare occurrence – people tend to discount small risks, and the large profits available from bot-herding outweigh the remote possibility of a couple of years in prison. Indeed, if you live in Ruritania, the chances of being collared are near-zero.

Another danger is that your botnet will be discovered by one of the white-hat bot-hunting groups, and your command and control (C&C) will be cut. The C&C is the means by which your bots report back to you and get their marching orders. As a couple of examples, this might be via an IRC channel or nickname (with the advantage of the anonymity they confer – thus reducing the risk of two years eating porridge [*widely believed to be the staple diet in British prisons*] – as well as the ease of moving to a different IRC server at a moment's notice), or it might be via the URL of a website. The C&C channel is the botnet's weakest link, and these white-hat groups hunt bots voluntarily, just because it's the 'Right Thing To Do' (and fun).

MAINTENANCE

So how do you maintain your botnet C&C when an opponent is trying to dissolve it? Some of the methods used include the use of free dynamic DNS services. Alternatively, you can use throw-away domain names and hosts. This allows for a quick response, moving from an at-risk IP address to a new one, while still using the same DNS record for the bot to connect to (only now, it would be pointing at a new IP). In the same way, the DNS records can be discarded and replaced while the IP address remains constant. Alternatively, you can point the host at many different IP addresses, or use many different hosts.

Botnet C&Cs also enjoy the benefits of redundancy, with unlimited fast-changing IP addresses and hosts ('fast-flux'), and failover capabilities with alternate C&C channels in case the first ('the head of the hydra') is compromised or fails. Further, a botnet may operate much like a terrorist cell, where different botnets are used to operate each other in a tree-like structure. For example, botnet A consists of 10 computers, and each bot within that botnet controls 100 computers contained in botnets B–G, and so on, with each of the branches compartmentalized from the others.

But these are today's botnets – it is possible to design a botnet that doesn't have the weak link of a C&C, and maybe

that will happen in the future. At the moment, whenever a C&C becomes more complex, it also becomes that much easier to detect. As a result, today's miscreants are focusing on making the take-down of the C&C channel irrelevant.

THE MONEY GAME

Finally, we reach the name of the game – profit! The return on investment from running botnets is significant. The people who run botnets can use them for any purpose. Although within the miscreant community there is a full social structure with many players, most botnets are run by organized crime groups involved with phishing and other types of financial fraud. That said, some botnets are still run by kids with nothing better to do. There is enough profit for everyone.

The uses for botnets are endless, varying from click-fraud and spyware installations to phishing and credit card theft, with identity theft, spam, anonymity, the launching of distributed denial of service attacks (often with the intent of blackmail) and malware-spreading thrown in for good measure. Botnets are weapons, and while they can be used for surgical strikes and intelligence gathering, they can also be utilized as weapons of mass destruction. Once the trojan is installed, it is only a matter of a small change in the source code or new instructions in order for it to be optimized for different uses.

Among the most difficult types of attack to deal with are distributed denial of service (DDoS) attacks. A target site is hit for an hour, by 100,000 different computers all over the world. It is extremely difficult and often impossible to distinguish their access attempts from accesses by genuine, money-spending customers. The site is brought down by the sheer impossibility of servicing all these accesses. When the DDoS starts up again the next day, the owner of the site is wits-ended, and if offered an end to his problems for $5,000, is likely to be very tempted to pay up.

But, of course, once you pay the Danegeld, you never get rid of the Dane. The $5,000 would only be the first instalment. Further, who is to assure you the miscreants won't attack regardless, or that others won't soon follow?

Which brings us to the classic problem that any fishpaster faces (blackmail is such an ugly word): how do you collect the loot, without being collared? The same problem faces phishers. Your botnet won't help you ... or will it? Enter the mule. A mule is someone who carries something of value, and who also carries the can if anything goes wrong. Mules are recruited by sending out spam using botnets. You've probably seen some of these spam messages amid the flurries of advertising for pills, potions and penis enlargement. 'Make money from home.' 'Join our international financial agency.' 'Become an export agent.'

Here's how it works: the mule receives the money and pays it into their bank account, keeping 10% for themselves, and sending the rest onwards using, for example, *Western Union* (which is a good way to receive money because the pickup is anonymous). The only purpose of the transaction is to muddy the money trail, launder the cash. Any problems, such as the FBI turning up at the door, will affect only the mule – and there are plenty more where he or she came from. Of course, the mules aren't told that – 'No risk', the recruitment messages say.

THE PROBLEMS

The bot-herders' C&Cs have become so protected with redundancy and secondary control channels that taking these down no longer has any kind of effect on the problem – other than pushing the miscreants to work harder at developing new technologies, and having them use alternative ISPs.

And indeed, the technology has advanced significantly and steadily over the past years, from simple IRC-based trojans in 1996–7 to today's fully fledged DNS control-based trojans using man-in-the-middle rootkit technology, effectively adding quality where before this game was about quantity alone. These listen in on every HTTPS session to steal credentials, aiming for financial information. Indeed, one of the main businesses of botnets today is stealing credentials and information wherever they manage to infiltrate. On an operational level, the organized groups behind the botnets operate dedicated teams dealing with everything from stolen data and deciding which stolen accounts are worth their time, through money transfer, all the way to real-world operations to support their criminal activity globally.

With a return on investment amounting to tens of thousands of US dollars for a relatively small botnet with click-fraud alone, and with damages from phishing alone likely to reach as much as two billion US dollars in 2006 globally, the bad guys are not likely to change their occupations any time soon.

Much as with every other established threat (which was ignored when it was small enough), it is a never-ending arms race of small victories on both sides as each escalates with a new technique or technology, forcing the other to adapt.

Leaving other concerns aside, millions of identities are being stolen every day across the world, through the use of the man-in-the-middle trojans utilizing rootkit technology. Organizations ranging from a moms&pops shop to Fortune 50 corporations have compromised machines on their networks, which are sending out spam or participating in DDoS attacks. These can easily be found by anonymous third parties and utilized for espionage.

Data stolen from a company's clients often affects the organization too. As an example, stolen credit card

information affects the credit company as well as the business from which things are purchased. The problem is: how do you secure something when it is on the remote client side, completely outside your control?

FUTURE

That's today, but what of tomorrow? Here are just a few examples of what we can expect to see in the not very distant future:

- Information already extracted today could be better analysed and used for more advanced purposes. Organizations with bots on their networks could be targeted for industrial espionage. Further aggregation and correlation of data could be introduced, so that financial attacks evolve to intelligence-gathering beyond stealing money or hijacking on-going transactions. The world will slowly be mapped in advanced social networks, seeing who is in business with whom and their level of financial ability.
- Strength of arm, which already rules the Internet, will become even more apparent, with spammers and organized crime groups protecting their business interests, causing damage and hurting businesses and public alike when they are threatened.
- Reputation systems will become even harder to implement when it is very difficult to establish whether a user identifying to a service is a genuine user or a bot. Trojans, i.e. bots, will transform from ill-behaving entities online to advanced critters that simulate regular user behaviour, making their detection extremely difficult.

Perhaps most importantly, there is little in place to change that. Law enforcement organizations cannot deal with the immense number of complaints they receive daily, and very few of them have the expertise to handle these cases. When they do, other parts of the legal system are fearful of computer-related investigations and will avoid them if at all possible. They are heavy-duty, demand a large investment of resources and are very technical, to a level that is often extremely confusing. Furthermore, a murder case is more sexy. Investigations that require wire-tapping, long-term research and global cooperation are even less likely.

The lives of the law enforcement organizations are not easy, though. We work with many skilled, able and smart folks. They want to help. Earlier we mentioned that bot-herding is an illegal activity. That isn't always the case – the law often tends to be one step behind current events, especially where technology is concerned, and in some countries, bot-herding isn't illegal. Furthermore, being illegal does not make it actionable to law enforcement – how do you prove damage from running a command and control server?

SOLUTIONS

On the network side, ISPs are left with the choice of protecting their networks on their own, which may, in the case of law enforcement interest, hinder investigations. The bad guys adapt, change their IP addresses and host names ever faster, and become very difficult to stop. In the past year there has been an on-going migration of C&C servers to China.

Anti-virus products detect samples and in some cases remove infections, yet with an average of 12,000 new bot samples coming out every month, around 10 per cent of which are financial fraud specific (and most of the remaining being multi-purpose trojan samples), the traditional anti-virus solution, as important as it may be to this fight, is no longer up to scratch as a lone solution which is inherently reactive.

Cooperation between ISPs (which in some cases run honey nets, unwillingly host these malware and phishing sites, etc.) and anti-virus vendors (who see what the samples do and what C&C channels they connect to) through operational vetted groups such as DA (Drone Armies) and MWP (Malicious Websites and Phishing) helps to mitigate some of the problems.

Botnets are a serious problem, but this is merely an example of a much larger problem with Internet security today. It is an economic issue, and without an economic solution that changes the miscreants' cost vs. benefit equation by reducing their gains and significantly heightening their risk, not much will change. Every new technology invented will be countered or circumvented by moving to new attack mediums, spam being a good example.

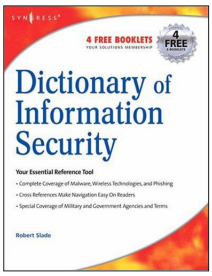
The problem will not go away, but the bad guys behind it can be put under more stress so that it becomes manageable. Law enforcement needs assistance, and not only with more resources. Policy makers should set the pace, allowing the law enforcement bodies to handle these cases to begin with. Further, extradition laws around the world can really come in handy. Some international work to recognize computer crime for what it is and tip the balance would be very helpful.

Every party in this fight is busy, and has their own business to take care of. That said, without better cooperation, intelligence gathering and coordinated response online as well as in the physical world, we believe that the current threats will eventually become unmanageable on the infrastructure level. Get involved with the vetted operational groups mentioned, meet some of those involved and see how you gain information to help your business, get introduced to the latest threats that others outside of your field see first, which will also affect you, and help to turn the tide back, fighting the real, original enemy (which is not the marketing department or your competitor). You can find more information about these groups at http://isotf.org/.

BOOK REVIEW 1

WAR OF THE WORDS

David Harley
Independent author, UK



Title: Dictionary of Information Security
Author: Robert Slade
Publisher: Syngress
ISBN: 1-59749-115-2
Cover Price: $29.95

Although Robert Slade's *Dictionary of Information Security* has only just made it to the printed page, it replaces his online security glossary, which for several years resided at http://victoria.tc.ca/techrev/secgloss.htm. (The glossary has now been removed, but the page remains as a home for errata and updates to the printed dictionary.)

Slade's credentials in the security field are impressive, as a writer, book reviewer and instructor. In fact, this book derives in part from his professional involvement with *(ISC)2*, whose Common Body of Knowledge (CBK) is the basis for the CISSP qualification. The web version of Slade's glossary was a popular free resource for CISSP candidates, and will no doubt be missed.

Glossary compilation in this area is a complex and frustrating task. The security field is knee-deep in obscure, inconsistently used jargon. Even worse, individuals and groups go to extravagant lengths to invent their own terminology, ignoring perfectly serviceable 'not invented here' usage. It is not easy to produce definitions that are reasonably short, clear, accurate, and which don't rely on an assumed knowledge of esoteric terms and concepts. Both the CBK and Slade's dictionary attempt to address these problems by introducing a consistent source of baseline definitions.

TARGET AUDIENCE

The cover notes and the author's preface suggest that the book is appropriate for security professionals and specialists, CISSP and other certification candidates, students of computer science or computer security, system and network administrators, and managers with security responsibilities.

STRUCTURE

The book contains no fewer than five forewords, each by a well-known and long-established name in information security and assurance: Fred Cohen, Jack Holleran, Peter G. Neumann, Hal Tipton and Dr Eugene Spafford. In addition, there are short biographies of the author and foreword contributors, publisher and author acknowledgements, plus a preface and an 'Introduction to Infosecspeak' by the author.

Does a relatively short dictionary actually need five forewords? Perhaps not. However, the fact that so many acknowledged experts are willing to contribute says something about the author's standing in the field.

The book is quite short, given the breadth of its subject matter: the main body runs to 222 pages, including the appendices. However, according to the author, the book's objective is to cover 'all the basic jargon of security, without bloating itself with every minor variation on a terminological theme'. The Preface and References sections include pointers to a range of alternative resources for those who need more detail in specific areas. (It's always a pleasure to read a security book whose author doesn't assume that no reader will ever need to consult another information resource.)

Unsurprisingly, the book follows a straightforward dictionary format (though there are no notes on pronunciation or, in general, etymology): a section for each letter of the alphabet, plus sections for symbols and numbers, which happen to contain one item each – '*-property' and '3DES'. There are, however, two appendices.

- Appendix A is a references section: rather than attempting to supply references for each entry, the author simply lists (with a short evaluative description) a number of communications-related dictionaries, glossaries and encyclopaedias.
- Appendix B is an extract ('The Lagos Creeper Box') from the fictional story *Stealing the Network: How to Own a Continent* (also published by *Syngress*). It is included on the grounds that the security risks to which the book refers could qualify it for a place in a security awareness program. This extract reminded me a little of the *Net Force* Tom Clancy franchise offshoot, albeit with added techie cred. Not without interest, but it sits oddly in the context of a security glossary.

Though much of Slade's previous writing is malware-related, this book is by no means virus-heavy. In fact, the malware content, albeit accurate as far as it goes, seems oddly dated. A number of older malware examples get a mention, but very little more recent than Nimda or Hybris. I agree that it would be counterproductive to try to include the name of every virus that the reader may have heard about. However, it seems odd to mention more-or-less extinct malware such as Michelangelo or Jerusalem, but to omit more recent high-profile malware such as Sobig and MyDoom. Similarly, there is no specific reference to botnets, specific bots (though zombies get a mention), or to major network worms like Slammer and Blaster. It would improve the book to include a few more recent, high-impact

examples, or even to restrict the number of examples and include only those with a really high profile. There are definitions of phishing (and even of spear phishing), pharming and identity theft, but not of money-laundering or mules (or even of puddle phishing). However, the author points out that this is very much a work 'in progress', anticipating ongoing updates and further editions for years to come. He even includes a pointer to a mailing list for anyone wanting to help with the project, so it seems likely that such anomalies will be dealt with in due course.

DOES THE BOOK KEEP ITS PROMISES?

The *Dictionary of Information Security* is well written, clear, and while no two security experts are going to agree on *every* aspect of *every* definition, accurate. The tone is informal and commendably anti-jargonist. Some of the entries are more flippant than others (check out Ohnosecond, the Ninety-Ninety Rule and Wannabe), but I found that rather refreshing.

A reasonably computer-literate general reader might find it a more consistent and accurate guide than most web resources, without being overly technical. It should find a ready market among computer science and information security students, and even more so among security certification candidates. It would be particularly useful to CISSP candidates to supplement the 'Official (ISC)2 Guide to the CISSP Exam'.

Security professionals needing a definition outside their own speciality may find it a good starting point, and the seasoned generalist might find it useful sometimes as a reliable memory jogger. However, I see it as being more useful to those unfortunate souls who find systems security administration or management thrust upon them suddenly, and who are struggling to keep their nostrils above the water line.

Most of all, it will be appreciated as a source of dependable baseline definitions by anyone who has learned to mistrust the astonishing volumes of misinformation that appear when summoned by *Google* searches on security terms.

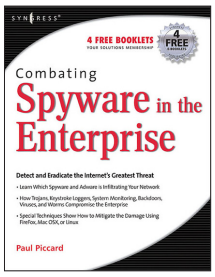
The editing and proofing is generally to a high standard, though there are one or two loose ends: for instance, the definition of ItW refers to the WildList, but there is no definition of the WildList or the WildList Organization. EICAR gets a mention, but CARO does not. URLs are not generally included, which makes sense: it's much less painful to maintain a resource that is impervious to the whims of webmasters. However, definitions of items such as BS7799 and ITIL might benefit from specific information on where to find reliable further information.

Slade's book fills a pretty wide gap in the market, and is highly recommended.

BOOK REVIEW 2

I SPY

David Harley
Independent author, UK



Title: Combating Spyware in the Enterprise
Author: Baskin, Bradley, Caruso, Faircloth, James, Piccard, & Schiller
Publisher: Syngress
ISBN: 1-59749-064-4
Cover Price: $49.95

According to the cover blurb, this book is essential reading for 'anyone responsible for the security of an enterprise's network'. It contains some useful and interesting general material, but does it live up to its claim?

CONTENT

The book begins with chapters entitled 'An overview of spyware' and 'The transformation of spyware', both written by Tony Bradley. The first defines spyware, malware, adware, parasiteware (browser hijackers), phishing and botnets. The definitions will not add much to the knowledge of readers of *Virus Bulletin*, but are uncontentious and clearly written, with examples of specific programs and the body text of several phishing emails.

A short description of how botnets work is followed by *very* short descriptions of a handful of bots. The separate section on malware seems a little odd, given that most of us would probably consider most of the programs described here to be malware. The second chapter is largely historical, describing the origins and evolution of spyware through targeted marketing, spam and cookies, and adware. A section on spyware and criminal activity introduces some slightly different or additional definitions (identity theft, ransomware) and is followed by a short US-centric section on anti-spyware legislation. While these chapters don't really address the enterprise context, they do provide a reasonable introduction to the topic of spyware.

Chapter three, 'Spyware and the enterprise network' by Jeremy Faircloth, begins with brief descriptions of a selection of hardware and software keystroke loggers, including *Sony*'s DRM fiasco. A consideration of 'spyware/backdoor combinations' and 'encapsulated trojans' is followed by a couple of pages on fake removal tools. The content related to the enterprise network is sparse and very generalized (e.g. 'Always use standard security practices…').

Chapter four, 'Real SPYware – crime, economic espionage, and espionage' by Craig S. Schiller, picks up the pace

somewhat. The first few pages consist mostly of historical overviews of the criminal use of (loosely speaking) spyware and commercial and governmental espionage, and seem to suggest that profit-driven malware represents a shift from a previously ethical model of virus writing. (I'd love to hear that debate at a *VB* conference!) A more detailed overview of phishing is followed by a long section on botnet functionality, detection and countermeasures. There's some useful introductory-to-intermediate material here, though many enterprises will not have the resources or incentive to follow up on this material to the same level of detail.

Chapters five and six, 'Solutions for the end user' and 'Forensic detection and removal', were written by Brian Baskin. Home users might find the former quite useful. However, this chapter is surprisingly long for a book which supposedly focuses on the enterprise. Only one keylogger detection utility is mentioned, but a number of common toolbar utilities are named, as well as a few commercial solutions. However, these are considered from an individual PC user's viewpoint, rather than in terms of enterprise management. Of the mainstream security vendors with products or services that include spyware management functionality, only *McAfee AntiSpyware* gets a mention. Given the number of mainstream AV vendors with a foot in that door, this is disquieting.

Chapter six is useful, but inaccurately named. It considers detection of spyware by tools like *Hijack This*, examination of the Registry, processes, the hosts file and so on, but pays no significant attention to the presentation of evidence in a court of law, so in what sense is it forensic? Its juxtaposition of detection and removal techniques without even mentioning the need to preserve a chain of evidence is, if anything, anti-forensic. The final section of the chapter summarizes a handful of enterprise-level removal tools and services, but not in any great depth.

Chapter seven, 'Dealing with spyware in a non-Microsoft world' by Ken Caruso, addresses the general issues of spyware and security on the *Linux* and Macintosh OS X platforms. Caruso mentions the existence of *Linux* spyware and rootkits, but the only *Linux* threats he describes (briefly) are Staog and Slapper, and the only preventative measures mentioned are the use of unprivileged accounts and (in the summary section) *tripwire* (which isn't described). Pre-OS X malware isn't mentioned at all, but Leap and Inqtana are described briefly. The only Mac security product mentioned is *MacScan*.

Chapter 8, 'The frugal engineer's guide to spyware prevention' by Paul Piccard, contains reasonable basic material, mostly on application security. It seems unhelpful to mention free versions of commercial AV here: very few enterprises will meet the licensing criteria to allow them to use those versions. The descriptions of *Microsoft*'s *WSUS* and *MBSA* and the sections on securing email, *Windows* and so on could be the starting point for a useful set of checklists, but leave a lot of ground uncovered.

The appendix, written by Lance James, contains some competent material on mule-driving, telephony, and malware trends. It does fit quite well with the heavy emphasis on phishing in other chapters, but doesn't really tie the subject in with the main theme of the book.

DOES THE BOOK KEEP ITS PROMISES?

This is a disappointing book. It contains useful general information on spyware and a number of related areas (especially phishing), but it isn't the definitive work on spyware. While there are certainly links between phishing attacks and spyware, the terms are not so interchangeable as to justify the volume of non-technical phishing material. This would have been more defensible had there been more emphasis on corporate governance and non-technical countermeasures. I would expect a book centred on spyware in the enterprise to address topics around governance issues like policy, end-user education, top management buy-in, compliance issues and accountability, as well as purely technical matters. Even at the technical level, the book is much better on attacks than on countermeasures.

The book largely overlooks the strong presence of mainstream AV vendors in this space. More surprisingly, even the open source programs widely used as a supplement (or, more contentiously, as a substitute) for commercial AV are not considered. This is a pity: a responsible, well-informed discussion of when it is appropriate to use open source and freeware would have been a real service to the enterprise community. AV aside, the range of commercial solutions that *is* considered is astonishingly narrow.

This emphasis on in-house technical measures and cost-cutting misses an essential point about enterprise security. Many enterprises prefer to spend serious money on commercial products and services rather than rely on internal expertise and applications that aren't contractually supported. Why would they do that? Because the principle of transferring risk and accountability is, if properly managed, a viable security model. A book on enterprise security that doesn't give due weight to this model undermines its own credibility.

This book contains useful reading matter for non-specialists, and many system administrators and managers might benefit from it. However, as a guide to corporate handling of spyware, it is weak and even misleading.

PRODUCT REVIEW

KASPERSKY INTERNET SECURITY 6.0

John Hawes

Kaspersky's current home-user offering goes under the functional title of *Kaspersky Internet Security 6*, hereinafter referred to as *KIS6*. Aiming for the all-in-one, covering-all-the-bases, eggs-in-one-basket end of the market, *KIS6* is a fairly complete package, combining standard on-demand and on-access virus checking with web and email filtering, a firewall, anti-spyware and anti-spam functionality, intrusion protection, and more besides.

Since I began playing with *KIS6* for the purposes of this review, the product has been repackaged and given away, with some limitations on functionality, by the web giant *AOL*. What the future of this channel will be is unsure, but it seems likely to bring *Kaspersky*'s brand increasingly to the fore in the home market. To see how the product fared in the world of the everyday user, I put on my ordinary Joe hat, tried not to think too hard, and charged straight in.

DOCUMENTATION, HELP, SUPPORT AND ONLINE INFORMATION

The user guide supplied with the product is also available as a PDF download, but is not included in electronic form on the rather empty distribution CD. It opens with a fairly lengthy introduction on the general subject of computer security, malware and best practice, followed by clear and thorough instructions for the setup and use of the product. My only quibble with this in-depth document is that, at some point, the section referred to in the guide as 'Program Tools' has been renamed 'Service' in the product itself – which caused some confusion until I figured out what had happened. This sort of change cropped up in a few other places too, and was particularly noticeable in the online help, when a link marked with one title would lead to a page with a different one (e.g. 'General' linked to 'Settings').

Kaspersky's web presence is based at www.kaspersky.com. The home page displays the green packaging of *KIS6* and the slogan 'Red for threats Green for you'. The rest of the page is fairly clean and simple, with the exception of the rapidly animated online scanner area in one corner (this offering is given prominence on most pages). A knowledgebase of support information is easily found, with a section dedicated to *KIS6* where many likely product-related queries are answered plainly. Among the information provided are details as to why the full product is superior to the pared-down version offered free by *AOL* (web-scanning, heuristics, intrusion defence and restore

tools are all mentioned, along with the many fine-tuning options, as being good reasons to opt for the full, paid-up version).

More malware-specific issues are dealt with at www.viruslist.com, which claims to be 'the largest encyclopedia of malware'. It certainly has a wealth of entries, most of which have at the very least a list of the aliases applied by different vendors, but many have little further information available, and as a malware information resource it lags behind a few other vendor sites.

Once information from the online databases has been exhausted, troubled users are pointed to the company's 24/7/365 tech support service, which is available in four languages and included in the annual licence fee. I dropped a few email queries to my local offices, and generally got a friendly and reasonably helpful response within about 10 minutes.

INSTALLATION AND COMPONENTS

I ran the installer from the CD – although it is also available as a download in English, French, German, Dutch and, of course, Russian; the download size is about 13MB.

The EULA, at a quick glance, contained nothing too frightening, and after selection of the root folder I was offered a choice of installation types. 'Complete' (for 'integrated protection of your computer') installs all available software, while at the other end of the spectrum, 'Anti-virus features' covers virus protection only, leaving out all the other bells and whistles. In between is a 'Custom' option, which is recommended for advanced users.

Having virtually no previous experience of *Kaspersky* products, I jumped straight into the custom mode, and found myself faced with a tree under the title 'KIS6'. Beneath the lonely 'Virus Scan' branch was a second, entitled 'Protection Components' and marked with a promising '+'.

Although not a great fan of the particular flavour of *Windows*-installer menu system used, I quickly figured out that the virus scanner itself is compulsory, while just about any combination of the other components is possible. (Checking back, I found that the 'Anti-virus features' option covered the scanner as well as on-access virus blocking, including web and mail filtering.) I left everything selected, and looked forward to being protected from viruses in files, on the web and via email, along with 'Proactive Defense', 'Anti-Spam', 'Anti-Spy' and 'Anti-Hacker' modules to play with.

After a very zippy first install came a series of set-up choices, all of which were highly configurable. The licence can be applied using the non-standard approach of pointing it to a file on a local disk, which was useful in my test lab due to the lack of Internet access. The default, however, is to enter an activation code found on the CD sleeve, following which a key is drawn from the ether. When I later tried this method on a machine connected to the web, it had some trouble connecting to the site from which it needed to download the file.

The update settings range from fully manual to fully automatic, with a fine-tuneable time period selection in between, and also allow for updates from a wide range of non-standard sources. On the lab machines, ignoring the update prompt left a sombre red warning lurking around most GUI pages, to make sure I didn't forget that my signatures were out of date. When I installed to a web-connected machine, downloading the several months' worth of updates released since the product came out took around half an hour, following which a reboot was required.

On the next page, scans can be set up of 'Startup objects', 'Critical Areas' and the full machine. These can be adjusted in terms of time but not content. Next up is the option to password-protect your settings, again with the option to protect only certain actions. I went first for the full-on coverage to see how much it would hamper my tinkering, but used a rather skimpy password, which was not remarked upon. There followed a moment of file copying before we got into some more technical setup options for the 'Anti-Hacker' module (a combination of firewall and intrusion detection). Networks are divided into zones and given security ratings; the *Windows Firewall* is detected if present, and there is a choice of shutting it down and replacing it with *Kaspersky*'s firewall, or leaving it on and installing the new one in an inactive state.

Next, some network applications are grouped together and have rules applied to them. These rules are configurable, and it is recommended to disable DNS caching. 'Interactive protection' offers to provide a warning when dangerous or suspicious things happen – dangerous only for the novice, while the more experienced user is offered the tweakable 'suspicious' option featuring a 'training mode'. With the full set of options selected, a reboot is required, and we're done.

At every point during the installation things were reasonably clear and well laid out, with sensible default settings explained in a way which might occasionally daunt the average untrained home user a little, but which would be unlikely to be truly terrifying. For the more adventurous, a button marked 'Options' or 'Advanced' is never far away, leading to a playground of fine-tuning tools. So far, the only choice I felt was missing was the ability to customize the selection of items scanned on start-up. The process was also fairly slick and speedy – I suspect that, had I selected all the defaults on a machine with a good reboot time, the whole thing could probably have been zipped through in under a minute.

CONFIGURATION AND OPERATION

On initial boot, after some popups informed me that the product might be out of date, and that a full scan of my machine had yet to be carried out, I was asked for my password before I could activate the product properly, and was offered the option to save the password for the current session.

A red-and-black 'K' sat in my system tray. Right-clicking it gave me options to scan my computer or run a scan (which opens the *KIS6* GUI at the scan page), update, check a network monitor (which shows me tables of connections, open ports and traffic), or view the settings (which opens the clear and admirably thorough configuration page). There was also the bold 'Open KIS6' (the default for the button, leading obviously to the main product interface), 'Pause protection' (which stops on-access scanning, giving a choice of time periods before reactivating, ranging from one minute to never), and finally 'Exit', which shuts the whole thing down quickly and efficiently, accompanied by appropriately worried popups from the Security Centre.

The main GUI itself is simple and clear, with menus and warning boxes down the left and a summary screen on the right, adorned with simple cartoon-style logos: a green umbrella for 'Protection', a magnifying glass for 'Scanning' and a globe for 'Services'. Reassuring green check marks informed me that protection services were running and that no threats had been detected, while a more sombre red '!' warned me again that I really should update my signatures.

Each pane, representing a different facet of the product, provides a nice clear overview of the protection provided, with status and statistical information and remarkably fast-acting stop/go/pause buttons. The large 'Settings' button in the top corner of the screen brings up the configuration page for each module, with some simple controls and numerous 'Settings' and 'Customize' buttons to allow for tweaking. The clarity was as welcome as the depth of configurability.

VIRUS SCANNING AND BLOCKING

A few brief scans over the *VB* test sets were easily configured, and trundled along at a fair speed with little difficulty. I found the unearthly scream that accompanied a virus detection a little disturbing, especially as I had set the warning level to 'Alert me at the end of the scan' and was expecting it to run silently. Without intervention, the yelps came along approximately every 15 seconds during a scan of the full infected collection – about one for every 100 viruses found. This was one of the only pieces of functionality that I couldn't figure out how to switch off, and I was forced to leave the room for a while to get away from the noise. On my return, having resolved to dig deeper, I did manage to find out how to deactivate the sounds, thanks to a help page which featured some misleading terminology (telling me to visit the non-existent 'Tools' page) but also a useful link to the right place, where I found copious options to configure the squealing, along with email alerts, popup balloons and so on.

At one point I managed to upset the scanner by foolishly clicking the 'neutralize all' button halfway through the scan. This left me staring at an hourglass for over a minute while the product got round to changing its balloon from a warning box to one with options to disinfect or delete (or skip, along with an 'apply to all' box which, thankfully, allowed me to carry on with the scan). Stopping and restarting the scan invariably offered the chance to pick up where it left off or rerun from the beginning, which I found useful. The timer seemed a little temperamental, claiming already to have been running for 40 minutes only 10 minutes in – this soon righted itself however, once the 'Finish time' prediction that runs in conjunction with the (fairly accurate) progress bar had settled down to a stable estimate.

When running a scan of an entire (clean) machine, the product was nice enough not to slow down anything else going on – in fact, it slowed itself almost to a halt when anything else was running on the machine, and only really got going when it was allowed free rein of the available resources. Even in full-paranoia mode there was little effect on other operations, although the scan did take many hours on a slower, more well-used machine.

The product sailed through the *VB* collection, happily picking up all the WildList set and missing nothing in the zoo either, only seemingly to get snagged for a time on the very last file, overrunning its predicted finish time by several minutes. I soon realised this was a recurrence of the above problem, apparently going back over the vast swathe of infection reports to find the first one before presenting it

to me. The 'Prompt for action when scan is complete' option produces a dialog box at the end of the scan (or, in the case of a scan of a large area with many thousands of infected files, a minute or two later), and the timer for the scan continues, rather oddly, until an action is selected.

On-access scanning over the collection ran at first at a fair speed too, although trying to do anything else while this went on was painfully slow, as is perhaps to be expected with so many infections being found at once. Once it got above 5,000 infected files accessed by the same process though, things slowed to a snail's pace, and a job some scanners have managed comfortably in half an hour ran to well over six.

Opening huge numbers of uninfected files in a similar manner presented no such problem though, so presumably on an average home user's computer – which, one hopes, would rarely have more than a handful of infected files on it at any given time – this would not present a major problem; it's also possible that some kind of tweak to the near-infinite configuration settings would speed things up. Looking at it more closely, it seemed likely that the problem was something to do with alerts and counts of viruses – as the script ploughed through the infected files, the GUI trying to display stats of viruses found and blocked lagged far behind, hitting the 100 mark at about the time when 5,000 had, in fact, been stopped.

My only other issue with the on-access protection feature is the apparent absence of an exclusions list, although admittedly such a function is likely to be more useful to virus researchers and testers than to the average home user.

OTHER FUNCTIONALITY

Using machines connected to the web, or to spoofs of it, I was able to play with some of the other functions. The 'anti-hacker' segment seemed fairly happy in its training mode – it would occasionally let out its wild yelp and suggest I should be worried about programs using 'invader' techniques, or query whether I really wanted to let my browser connect to the web, but once it had learnt a few of the basics it seemed content that I wasn't doing anything crazy.

A spell of web browsing seemed to suffer no significant slow down, and every ten minutes or so I was informed of a blocked attack from some network worm or other, which made me feel very safe. When I started trying to download suspicious files the scream would be there to warn me, and nothing unpleasant slipped by its guard when I tried downloading the entire WildList from the lab webserver. Sadly, the anti-spam functionality fell outside the scope of this test, but the whitelist 'training' area requires an *Outlook* inbox containing a minimum of 50 legitimate emails, to acclimatize to your mail habits; mail anti-virus is independent of the mail program.

The anti-spy area includes phishing, dialler and popup protection, along with an anti-banner tool (not activated by default). Again, most of these have configurable whitelists and blacklists. 'Proactive defence' covers behaviour analysis, preventing suspicious actions such as injecting into other processes, along with monitors watching for changes to the registry and unwanted *Office* macro behaviour.

All of these tools have a nice sliding scale on their settings pages, running from switched off to total lock-down, with the recommended level somewhere in between, and with deeper and more complex settings available for the more advanced user, producing a reassuring sense of consistency.

On top of all this, there is also a command-line interface, allowing the serious techie to get his fingers dirty, and perhaps even integrate the software's varied functionality into other tools, and a boot-disk creation utility, which uses the *BartPE* utility (not supplied) and a *Windows XP* install CD to create a bootable live CD with *Windows* and *KIS6* all ready to be run, providing a fully secure safe mode.

CONCLUSIONS

With little to criticize and much to praise, this is a solid, well-designed product, neatly and seamlessly integrating a wide range of protection into a single, easy-to-use interface. Though perhaps not the fastest in terms of scanning times, especially during heavy on-access testing, this is perhaps the price one must pay for the thoroughness shown. The combination of easy-on-the-brain controls for the general user to ever more fine-tunable configuration settings for the expert is welcome and well thought out, and the visual design is a similarly happy blend of friendly, cartoon-like graphics with serious-looking controls.

Kaspersky has aimed to cover as many security aspects as it can in a single streamlined application, and to allow the user to apply them in as flexible a way as possible. In terms of clarity of design and simplicity of implementation, *Kaspersky* has created an extremely strong user interface, backed up by solid and thorough detection technology.

Technical details

The product was installed at various times on:

AMD K6, 256MB RAM, dual 10GB hard drives, no network, running *Windows 2000 Professional SP4*.

Intel Pentium 4, 512MB RAM, dual 20GB hard drives, 10/100 LAN connection, running *Windows 2000 Professional SP4* and *Windows XP Professional SP2*.

Intel Celeron notebook, 256MB RAM, 10 GB hard drive, 2Mbit ADSL modem, running *Windows XP Professional SP2*.

END NOTES & NEWS

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. Seven tracks of hands-on technical training sessions run on 18 and 19 September, followed by a two-stream conference on 20 and 21 September. Full details of the training sessions and conference programme, as well as online registration, can be found 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.

Mobile Security takes place 3–5 October 2006 in London, UK. The conference will include 12 operator case studies and a pre-conference workshop entitled 'Effectively securing premium content through interoperable DRM'. For more information see http://www.informatm.com/security/?src=vbn.

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. Register online at http://www.virusbtn.com/.

RSA Conference Europe 2006 takes place 23–25 October 2006 in Nice, France. Online registration and full details of the conference agenda are available now at 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/.

The International Conference on Human Aspects of Information Security & Assurance will be held 10–12 July 2007 in Plymouth, UK. The conference will focus on information security issues that relate to people – the methods that inform and guide users' understanding of security and the technologies that can benefit and support them in achieving protection. For more details, including a call for papers, see http://www.haisa.org/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

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

vbSpam supplement

CONTENTS

S1 **NEWS & EVENTS**

S2 **FEATURE**
Present and future phishing techniques

NEWS & EVENTS

A FINE, A CURFEW AND A TREASURE HUNT

Kicking off a round-up of some of the anti-spam penalties issued worldwide this month, the Chinese government has made an underwhelming impression with its first fine for spamming. Hesheng Zhihui Enterprise Management Consulting was fined a paltry 5,000 yuan renminbi (approx. £331) for sending 'bulk sent emails containing advertisements to Internet users since January'. The country's anti-spam regulations – introduced in March this year – state that commercial emails must be sent with the text 'AD' in the header and must contain opt-out options. The maximum fine for failing to do this is 30,000 yuan (approx. £1,989).

Next up is the equally underwhelming case of 18-year-old DoS spammer David Lennon – his penalty is a two-month curfew that will confine him to his home from 12.30am to 7.00am on weekdays, and between 12.30am and 10.00am at the weekend. Lennon was originally cleared of charges in November 2005 when a judge ruled that executing a DoS attack did not contravene the Computer Misuse Act (see *VB*, December 2005, p.3). However, the ruling was challenged by the Crown Prosecution Service and later sent back to the Magistrates Court.

Lennon – whose 5 million emails crashed the servers of the *Domestic & General Group* – initially faced the possibility of having to pay costs of up to £29,000, but the demand for the costs was later dropped. The judge told the court 'Even given his age at the time, this was a grave offence and caused serious damage, so I need to impose something to make him think again' – however it seems that the penalty Lennon now faces will barely impact on his life (other, perhaps, than allowing him a little more beauty sleep than his teenage contemporaries). The curfew has been arranged carefully so as not to interfere with Lennon's job at a local cinema and ends on the day before he starts college.

Finally, an update to the story reported on www.virusbtn.com last month about ISP mammoth *AOL* digging for treasure. The company obtained a court judgement allowing it to dig up the land of a convicted spammer's family, in a search for a stash of wealth it believes he has stowed away in the form of gold and platinum bars.

Davis Wolfgang Hawke was convicted of spamming offences last year and ordered to pay *AOL* over $12 million – but he has since disappeared, leaving behind only receipts for large amounts of precious metal. Initially Hawke's family insisted that the idea that he buried the stash on their land was ridiculous. However his grandparents – while still believing that the ISP is following a defective treasure map – have now agreed to allow *AOL* to search their land using radar and sonar equipment. It is thought that the threat of a legal battle with the mammoth ISP was sufficient persuasion. However, Hawke's parents remain resolute in their refusal to allow the company access to their property.

SPOT THAT SPAMMER

Last month, *McAfee* invited web users to spot the undesirable website, in an eight-question quiz entitled: 'Can you spot sites that cause spam?' The quiz presents consumers with pairs of websites, providing a screenshot of the home page plus details of the privacy policy of each, and asks contestants to judge which site in each pair 'respects email privacy'. On obtaining a low score in the test, the consumer is told 'Watch out! Your inbox might explode!' (which seems a little severe) – and of course, advised to purchase *McAfee SiteAdvisor*. The quiz is available at http://www.mcafee.com/spamquiz/.

EVENTS

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

Inbox 2007 will take place 31 May to 1 June 2007 in San Jose, CA, USA. The two-day event focuses on all aspects of managing, securing and using email responsibly. For more details see http://www.inboxevent.com/.

FEATURE

PRESENT AND FUTURE PHISHING TECHNIQUES

Sorin Mustaca
Avira, Germany

In recent months, the number of phishing attacks has risen at a worrying pace. Between August 2005 and January 2006 phishing sites hosted between 70 (August 2005) and 120 (January 2006) attacks against company brands. Unfortunately, the prevalence of phishing sites is still rising at the time of writing this article (June 2006).

In April 2006 the Anti-Phishing Working Group [1] detected the highest number of phishing sites in its history. Whereas in April 2005 there were only 2,854 unique detections, by April 2006 there were over 11,000. There are many reasons for the significant increase, but the most important is the fact that phishing is profitable for the infractors. The laws are still not clear in this area; in some cases they do not even exist, and in countries where they do exist, their application is limited or simply too slow. Another very important aspect of the phishing 'industry' is the complexity of the attacked sites. The simpler the website, the easier it is to duplicate and, of course, the more copies there are in the wild.

A phishing attack has, in general, three parts: bait distribution through email, a fake website which collects data, and the manipulation and use of this information by the infractors. We can analyse, detect and even stop the first two parts because we can see them, but we are unable to do anything about the third part, because it takes place in the criminal underground.

Some phishing attacks are so alike that they seem to have been produced with email and/or website generators. Others are so badly constructed that you ask yourself how they could ever fool anyone. However, the attacks that give us the worst headaches are distributed phishing attacks.

In this article I will describe how these attacks are constructed, why they are so hard to detect, and what I think will follow soon.

THE PHISHNET

A distributed application has a client-server architecture with two, three or *n* tiers. Distributed implementations are based on multiple levels of complexity, all of which are characterized by the distribution of processing logic. In the case of phishing, the bait emails represent the clients and the fake website represents the server. As can be seen in Figure 1, the phishnet is a three-tier distributed application. In such an architecture, the third component – the one that resides in the middle – is used to accept, process and distribute requests from various clients to the server.

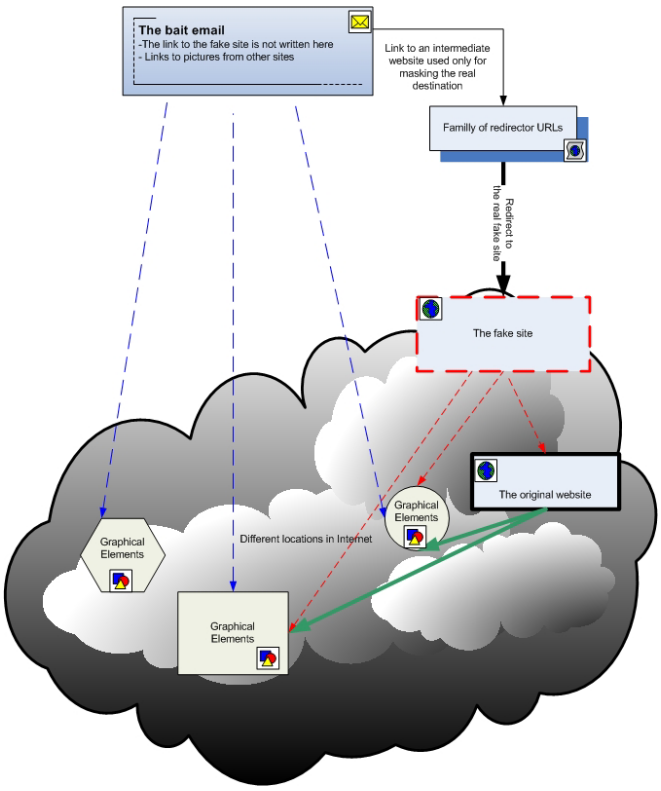


Figure 1. The phishnet today.

The middle layer is implemented by the phishers in a clever way. The bait email comes with a link to a website that has (almost) nothing to do with the phishing attack. It seems that some known vulnerabilities of *Apache 1.3.3x* are being used [2] to gain partial control over legitimate sites – enough control to allow phishers to place some files on the site. These files (which may be PHP or HTML) perform a redirection from the legitimate site to the phishing site. The phishing site hosts all of the files required by the fake site. Both sites use *Apache 1.3.3x*.

I have seen a couple of variations on this theme, where www.google.com or www.yahoo.com have been used to redirect to the fake website. In these cases the graphical elements of the fake site were taken from the legitimate website.

Phishers are becoming increasingly skilled, and they seem to know about all the things that anti-phishing products search for – such as double links. This is why we very seldom see double links like this any more:

```
<a href="http://fake_site.com">https://www.paypal.com</a>
```

and more often see links like this:

```
<a href="http://fake_site.com">Verify</a>
```

as well as numerous variations on this method (using images, tables, map areas etc.).

Of course, it is pretty easy to filter IP addresses or domains such as http://fake_site.com directly from the email client or from the proxy server, which is the reason why the redirect feature is so popular with the phishers.

Since mid-June this year, a new trend has emerged among bait emails. Instead of containing simple links, a bait email now contains a form with a button that redirects the recipient's browser to the target website. For example:

```
<form name="form1" action="http://fake_site.com">
<input action="http://" method="post" type="submit"
name="Submit" value="Remove Limitations">
```

Or

```
<font size="2" face="Arial, Verdana">
<INPUT style="BORDER-RIGHT: 0pt; BORDER-TOP: 0pt;
FONT-SIZE: 10pt; BORDER-LEFT: 0pt; CURSOR: hand;
COLOR: blue; BORDER-BOTTOM: 0pt; BACKGROUND-COLOR:
transparent; TEXT-DECORATION: underline"
type="submit" value="https://www.paypal.com/cgi-bin/
webscr?cmd=_login-run" tabindex="2"></font></a>
```

The result is a button with the value 'Remove Limitations' in the first example (where the body text of such a bait email may try to persuade the recipient that they need to take certain action in order to restore full access to their account), or with the official paypal.com website in the second example. Note that in both cases the button is barely visible, thus making the second version look exactly like a valid link. The interesting thing is that this trick works with all browsers except those that are *Mozilla*-based.

THE FUTURE STARTS NOW

From what we've seen so far, the success of phishing activity appears to depend on a number of factors:

1. How 'real' the email looks.
2. How quickly bait email is distributed, how many recipients read the email, and how many recipients click on the links inside before the mail is blocked and the site is shut down.
3. How quickly the fake site can be activated.
4. How 'real' the fake site looks.
5. How hard it is to detect the emails, and maybe the fake sites automatically (note that the fact that they 'look' real can be seen easily by a human and not so easily by a machine).

Factor number 2 is mainly a matter of luck and of the number of targets available. This in itself has created another industry; that of email addresses harvesting in order to provide the phishers with fresh email addresses. Phishers demand fresh email addresses because, presumably, they assume that someone who has been fooled once will not be fooled again, and someone who receives a lot of phishing emails will not be fooled either. Of course, most of us receive hundreds of identical spams and phishing emails every day. So, in fact this part has more to do with human nature than with phishing techniques.

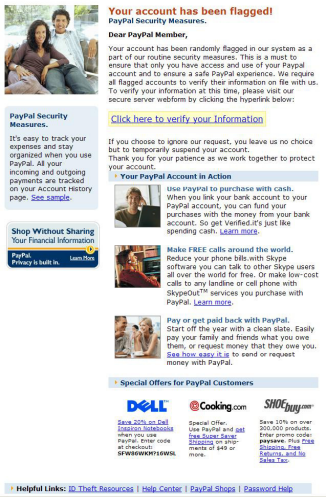


Figure 2. PayPal phishing email as real as the original.

THE FACTS

Figure 2 shows a bait email for a *PayPal* phishing scam. Some bait emails can look even more convincing than the genuine emails from the company in question. Combining this genuine-looking message with links that go over HTTPS protocol provides even more credibility to such an email [3].

I mentioned above that a website needs to be sent online quickly, and that it must be made to look realistic (points 3 and 4 above). This is where the saying 'a picture is worth a thousand words' really does ring true. The web page shown in Figure 3 is, in fact, an image of the real *PayPal* web page created as a *Macromedia Flash* program.

When I 'browsed' the fake website, none of the links worked – presumably because they were part of a screenshot taken from the real *PayPal* website. The only part of the page that 'worked' was the 'Log In' button once an email address and password had been entered. No validation was performed of the email address and password, so after the user has entered any string into the these fields, he can 'log in' and enter his credit card information – during this process, all the fields are validated (credit card number and type, ZIP code, etc.).

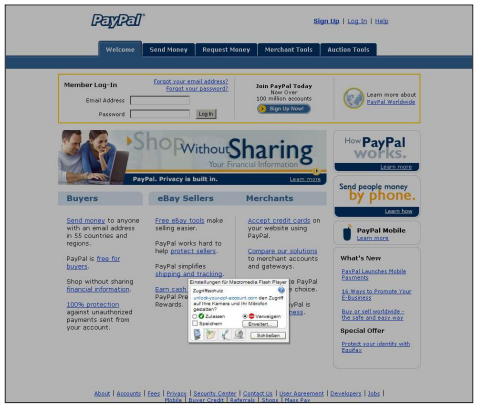


Figure 3. Website written entirely in Flash.

Imagine how quickly such

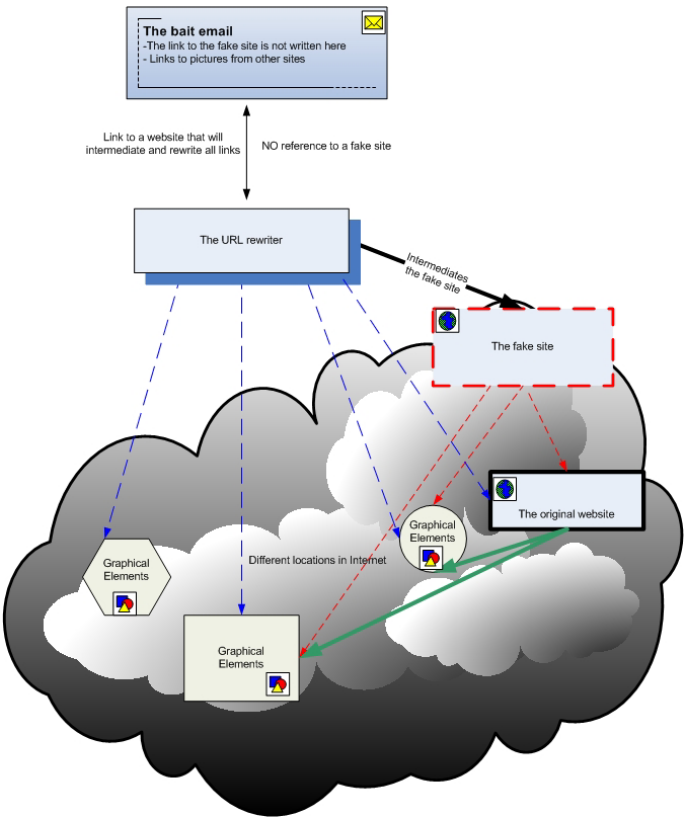


Figure 4. The future of phishnet.

a website can be published on a new URL. It probably has only one or two files: the *Flash* program and the database containing credit card information. There is no need to worry about pictures, Java Script files or CSS files.

Finally, in relation to point 5, there is one obvious way to stop a phishing attack: block the phishy website.

We all know of the anti-phishing toolbars marketed by various vendors (*Netcraft* [4], *Microsoft*'s *IE7* toolbar [5], *Google*'s *Firefox* extension [6], etc.) which use blacklists of URLs to prevent the user's browser from loading anything from the blacklisted location.

Imagine a phishing attack like the one described in Figure 4, where the bait email doesn't contain a link to the real and/or fake website but instead uses a URL rewriter. This technique will not be detected by the anti-phishing toolbars and it might bypass some anti-phishing mail filters as well (since the email contains no double links). This technique, which allows one to visit a website anonymously, is not new and it is available today on the Internet. Just google for 'anonymous surfing'. Fortunately, I have not seen any phishing attacks that use this method so far – currently it exists only for search engine redirects.

Such an attack might even contain unique links for every email sent, because the URL rewriter is based on a session ID. So, the email would contain links like 'http://url_rewriter/<unique id>/<file>', making it seem more legitimate than the official emails. The only issue here is that the bait emails will continue to come from someone@site.com and targets will not go to site.com.

CONCLUSION

As proven continuously by statistics, phishing is a very profitable (illegal) business, and one in which we see a lot of effort being invested every month. Phishing attacks are becoming more dangerous than ever because they have started to become very hard to detect automatically. We can analyse the emails and categorize them as good or phishy, but the major problem is where to draw the border between them. Nowadays, official emails often point to external information, meaning that they contain a lot of external links – which makes them look very similar to the phishy ones.

What can be done? For a start, we can ask those companies that are attacked to change some of their practices to make sure that they are not an easy target for the phishers. This means that they have to change the way they make newsletters, to be careful about what kind of information is contained in them, and maybe even to change some parts of their websites. In time, if applied correctly, this might make phishing attempts useless, but I doubt it.

ACKNOWLEDGEMENTS

Beside the references below, I would like to thank Oliver Auerbach and Cosmin Luta for their valuable suggestions.

REFERENCES

[1] Anti-Phishing Working Group, http://www.apwg.com/.

[2] Apache Vulnerabilities, http://secunia.com/product/72/?period=2006#statistics.

[3] http://www.avira.com/en/threats/section/fulldetails/id_vir/2099/paypal_58.html.

[4] Netcraft Toolbar, http://toolbar.netcraft.com/.

[5] Microsoft IE7 with anti-phishing toolbar, http://www.microsoft.com/windows/ie/ie7/about/features/default.mspx.

[6] Google's Firefox Toolbar with anti-phishing extension, http://www.google.com/support/toolbar/bin/answer.py?answer=34798&hl=en.