JUNE 2005

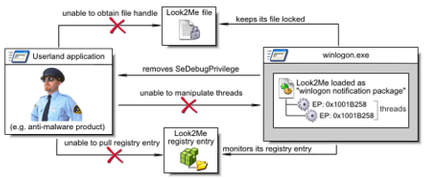
virus BULLETIN

The International Publication on Computer Virus Prevention, Recognition and Removal

CONTENTS

ISSN 0956-9979

IN THIS ISSUE



LOOK2 THE FUTURE

While AV vendors continue the tricky process of determining what should and should not be detected, adware is becoming increasingly advanced – both in the way it hooks the system and in the way it prevents itself from being removed. Sergei Shevchenko presents *VB*'s first adware analysis.

page 4

WINDOWS XP COMPARATIVE

This month's testing process proved to be relatively plain sailing for *VB*'s resident reviewer Matt Ham. Find out whether it was such a breeze for the 28 products on test.

page 11

vbSpam supplement

In the *VB Spam Supplement* this month: anti-spam news & events and Bayesian classification applied to phishing fraud.

'When will a silver bullet come along that makes computers work as well as toasters?'

John Aycock
University of Calgary

YOUR COMPUTER IS TOAST

Earlier this year, I was asked a question: how do you stop viruses and worms altogether? Completely. Full stop. No more viruses and worms any place. I had to think about this for a moment. It's a very interesting question, and my answer was somewhat surreal: toasters.

I love my toaster. From a user interface point of view, it's brilliant. Even my youngest child can understand how to operate it: it has few controls, and it's easy to form a mental model of how it operates. What's amazing is that – apart from the odd piece of burnt raisin bread – it just works. It's never required an update or a patch. And my toaster has never been hit by a virus or worm, nor has spyware ever absconded with my toast preferences.

The same claims cannot be made for any computer I've connected to a network, no matter what the architecture or operating system. Given how much our society relies upon computers, you would hope that the computers running the power grid were more reliable than the toasters plugged in to it. Yet it's no secret that our computers are breeding grounds for all kinds of malicious software. With mobile phone worms spreading in the wild, virus-like behaviour being exhibited by Sims 2 hacks, and proof-of-concept PDF file worms, is there any logical limit to the places where malware can thrive?

In *Profiles of the Future*, Arthur C. Clarke famously wrote that 'any sufficiently advanced technology is indistinguishable from magic.' I have a corollary to this, which I'll modestly call Aycock's law: any sufficiently advanced technology is susceptible to viruses.

Already we need anti-virus software on our desktops, laptops, and mobile phones; anti-virus for game machines probably isn't far off, either. When will a silver bullet come along that makes computers work as well as toasters?

One of the problems is that computer scientists like to generalize. A general algorithm is cleverer than a less general one; a general design is better than a more specific one. Our computers are general-purpose, and we interconnect them in the hope that they can talk to everything else in some general way. Call me a Luddite, but maybe I don't need my wristwatch chatting with my running shoes via Bluetooth. We don't require generality in every situation, and in some cases we are better off without it. For example, it's hard to verify the security of a web browser that's general enough to be extensible. The plug-ins that extend the browser aren't known until they run, which provides a lot of leeway for malware to exploit through software engineering and social engineering.

Computer memory is generalized, as something which can hold code and data, rather than code *or* data. This fact has been exploited by high-profile worms with buffer overflow attacks for over 16 years now, with the Internet worm in 1988, Slammer in 2003.

Worms, of course, can't spread across communication channels that don't exist. My toaster is not general enough to communicate with the blender beside it. However, the Internet has proven to be a general medium over which disparate devices can talk to one another. You can even buy Internet-enabled refrigerators, presumably to send spam as well as keep it chilled.

At the opposite end of the spectrum lie domain-specific systems. These are tailored to one narrow area, like SQL being used to describe database queries instead of using a general-purpose language like C. Toasters are domain-specific systems too, tailored to the domain of making bread brown. Domain-specific systems have two important properties relating to malware: their functionality is limited, and their normal behaviour is well understood. Suitably limited functionality can deny would-be malware authors from expressing their progeny, and well-understood behaviour allows extremely accurate anti-virus heuristics and emulation to be developed.

That's it. Design computers to do one thing, and only one thing, well. Resist the urge to have them communicate with all their neighbours within earshot. By limiting generality and unnecessary communication channels, hopefully Aycock's law is one that is made to be broken. Toast, anyone?

NEWS

SUN, SEA AND SECURITY

Photo courtesy of Eddy Willems.

The historical island of Malta was the setting for the 14th Annual EICAR meeting last month. With the sun beating down and an ocean view from the breakfast table, it took a little will power to venture inside for the start of the conference, but a well planned two-stream conference programme with some accomplished presenters proved to be plenty to keep delegates engaged for the two days of the conference. The full report can be found at http://www.virusbtn.com/articles/virusbulletin/conferencereports/2005/06_03.xml.

MICROSOFT CARE

Microsoft unveiled its new security service for consumers last month. Known as *OneCare*, the paid subscription service will provide anti-virus, anti-spyware and personal firewall protection, as well as offering PC maintenance services such as disk defragging and file repair, and scheduled data backup to CD and DVD. *Microsoft* describes the service as being designed 'specifically for people who don't have the time or technical expertise necessary to secure and manage a computer on a daily basis.' However, officials have indicated that, once the new service has been delivered, the company will turn its attention to developing an enterprise anti-virus offering.

OneCare is currently being tested by *Microsoft* employees, and is scheduled for public beta availability later this year.

PATENT SQUABBLES

An International Trade Commission (ITC) judge has recommended that *Fortinet* be prohibited from importing its *FortiGate* anti-virus firewall appliance products into the US after finding that the company has infringed a patent held by *Trend Micro*. US patent 5,623,600, which covers server-based anti-virus technology, was filed by *Trend Micro* in 1995 and assigned to the company in 1997. *Fortinet*, which contested the infringement claim, has announced that it intends to seek a review of the ruling. In a statement *Fortinet* founder, president and CEO Ken Xie said '*Fortinet* believes that it has been developing products with its own proprietary technology. *Fortinet* has and will continue to operate with the belief that all companies should respect the intellectual property rights of others.'

Prevalence Table – April 2005

Virus	Type	Incidents	Reports
Win32/Netsky	File	30,208	66.95%
Win32/Bagle	File	4,567	10.12%
Win32/Bagz	File	3,542	7.85%
Win32/Mydoom	File	2,448	5.43%
Win32/Mytob	File	647	1.43%
Win32/Funlove	File	393	0.87%
Win32/Zafi	File	370	0.82%
Win32/Klez	File	359	0.80%
Win32/Mabutu	File	304	0.67%
Win32/Dumaru	File	279	0.62%
Win32/Lovgate	File	273	0.61%
Win32/Bugbear	File	170	0.38%
Win32/MyWife	File	167	0.37%
Win32/Pate	File	150	0.33%
Win32/Valla	File	122	0.27%
Win32/Swen	File	119	0.26%
Win32/Sober	File	108	0.24%
Win32/Mimail	File	107	0.24%
Redlof	Script	76	0.17%
Win32/Mota	File	75	0.17%
Win32/Fizzer	File	52	0.12%
Win32/Yaha	File	51	0.11%
Win32/Sobig	File	41	0.09%
Win32/Hybris	File	28	0.06%
Win95/Tenrobot	File	27	0.06%
Win32/Maslan	File	26	0.06%
Win32/Lovelorn	File	25	0.06%
Win32/Kriz	File	22	0.05%
Win32/Nachi	File	21	0.05%
Win32/Elkern	File	19	0.04%
Win32/Magistr	File	19	0.04%
Win32/BadTrans	File	18	0.04%
Others[1]		290	0.64%
Total		45,123	100%

[1]The Prevalence Table includes a total of 290 reports across 62 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

ADWARE ANALYSIS

STANDING THE PRIVILEGE ATTACK

Sergei Shevchenko
Symantec Security Response, Australia

The security and resource implications of adware – particularly in the corporate environment – are becoming an increasing concern for users. While AV vendors continue the tricky process of determining what should and should not be detected, adware itself is becoming increasingly advanced – both in the way it hooks the system and in the way it prevents itself from being removed. Here, Sergei Shevchenko presents Virus Bulletin's first adware analysis.

Can you imagine a world where the police force has no right to investigate and law courts have no right to pass judgement? You might imagine that it would feel like being a rabbit in a laboratory cage whose behaviour and habits are closely monitored – with no means of escape.

This comparison came to my mind during the analysis of the privilege attack, combined with the 'winlogon notification package' technique, employed by Adware/Look2Me.

A TOUCH OF THEORY

In order to run in the address space of *Windows Explorer* and *Internet Explorer*, it used to be pretty common for adware to install itself in the form of Shell extensions and/or Browser Helper Objects. Being registered as an in-process COM object for the shell/browser gave the adware certain advantages: process invisibility, the ability to bypass the firewall, direct access to the browsing session events, the ability to be up and running as long as the shell is alive, and so on.

Another legitimate in-process model is provided by *Microsoft* in the form of the 'winlogon notification package' (*Windows 2000/XP*), which provides the additional advantage of running the code under the System account. Another advantage relates to the ability to define the local security policy on a stage when the authentication package creates a new logon session. This allows the adware to shape the token that LSA creates for the authenticated user, and therefore affect the rights of all processes running under its account.

When the system starts, winlogon is launched before the system shell. To locate its own extensions, winlogon enumerates the subkeys of the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

It then loads the located extensions and uses their exports to construct the interface for handling the following system events:

Lock	Shutdown	Startup
Logoff	StartScreenSaver	StopScreenSaver
Logon	StartShell	Unlock

Winlogon extensions are also loaded and run in safe mode, just like shell extensions.

REVENONS À NOS MOUTONS (LET US RETURN TO OUR SHEEP)

When run, Look2Me drops its DLL component and installs it as 'winlogon notification package'. The filename for the DLL module is pseudo-random: it is composed using the filename letters of the files located in the %system% directory.

In order to register itself as winlogon extension, Look2Me creates the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\[Random Key]

It then creates the following values in the key to establish the interface with winlogon.exe:

- Asynchronous = 0x00000000
- DllName = '%System%\[DLL filename, which is random]'
- Impersonate = 0x00000000
- Logon = 'WinLogon'
- Logoff = 'WinLogoff'
- Shutdown = 'WinShutdown'

Note: [Random Key] is picked up by enumerating the subkeys of the following registry key randomly:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion

If it cannot figure out what needs to be used for [Random Key], it will use the string 'Guardian' for this key. Look2Me then registers the dropped DLL as an in-process COM object, by creating the following registry keys:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\ID
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\IDex
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\Implemented Categories
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\Implemented Categories\{00021492-0000-0000-C000-00000000046}
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\InprocServer32
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[random CLSID]\Version

Then, Look2Me enumerates all explorer.exe windows and sends them an 'enable' message. At the next step, it terminates explorer.exe.

As soon as the shell is restarted, the adware's DLL module will be loaded into its address space. At this moment, Look2Me runs within the shell as its extension – however, the next time the computer starts, Look2Me will run as a winlogon extension within its address space.

STRIPPING THE PRIVILEGE

Any process that is started under the Administrator account is supplied with a copy of the Administrator's access token, to inherit its rights and privileges. For security reasons, not all of the rights and privileges are enabled by default (for example, to prevent termination of critical processes – e.g. a user cannot terminate winlogon.exe in the task manager). However, a process running in the Administrator account may still adjust the Administrator's privileges with the AdjustTokenPrivileges() API.

To use a real world analogy, consider a policeman who presents his documents at the entrance to a restricted area. He is identified as a policeman and so is allowed to do many things that others are not allowed, or are not expected to do. He is supplied with a particular set of rights and privileges, that are normal for the police force. For example, he can monitor other people closely and even ask them to show him their documents. However, without a search warrant, the policeman is not allowed to search a house. For security reasons, this permission is not enabled by default, but the system gives the policeman the instruments to obtain it (e.g. via a court application).

To deactivate a userland Administrator process, Look2Me opens the Policy object on the local system and enumerates all accounts in the LSA Policy object's database that hold the SeDebugPrivilege privilege.

Then, it removes this privilege from every account that has this privilege enabled in its access token. Primarily this will affect the LSA account BUILTINS/Administrators, which has SeDebugPrivilege enabled by default.

After a reboot, no user-mode process that runs in the Administrator context will be able to enable the SeDebugPrivilege privilege. The AdjustTokenPrivileges() API still succeeds, but GetLastError() returns 'ERROR_NOT_ALL_ASSIGNED', which means that SeDebugPrivilege did not accept the attribute 'SE_PRIVILEGE_ENABLED'.

Returning to our real world analogy, no policeman entering the restricted area would have any of the privileges that are normal for the police force. SeDebugPrivilege is a critical privilege that is vital for the successful removal of unwanted software. Without this privilege, the user-mode process acts more like a guest, with no right to perform critically important actions.

SETTING THE CHILDPROOF LOCK

Look2Me locks its file by opening it in an exclusive mode, so that CreateFile will fail with a sharing violation. As a result, no user-mode process is able to open and scan it.

Generally speaking, if the process has the debug privilege then the locked file could be unlocked by duplicating and closing its handle.

One method of doing this is to enumerate all open handles with NtQuerySystemInformation and NT_HANDLE_LIST system information class, pass the located kernel object pointers to the installed kernel-mode driver, and let it provide the user-mode part of the application with the object names by utilising the ObQueryNameString() API.

Look2Me modules would be identified by locating adware threads and their addresses in the running processes. The file-type objects that are opened and owned by the parent winlogon process would also be known by the name that is returned by the driver. This would allow handles for Look2Me module files to be found, and then closed. This method of closing handles is implemented in the *Sysinternals* tools *Handle* and *System Process*. However, to duplicate a handle, the parent process needs to be open with the PROCESS_DUP_HANDLE access right, and that still requires the debug privilege.

Note that, without the debug privilege, the modules can be enumerated by utilizing the NtOpenProcess(), NtQueryInformationProcess() and NtReadVirtualMemory() APIs and by inspecting Process Environment Blocks and the obtained module lists. Remember that if the debug privilege is removed and the inspected process is the system

process, then RtlQueryProcessDebugInformation() fails with the DEBUG_ACCESS_DENIED status code returned. This API should not be used to enumerate Look2Me modules within winlogon.exe.

REGISTRY WATCHDOG

Look2Me then spawns several threads that are responsible for different actions. Some of them install monitors on the registry keys by using the RegNotifyChangeKeyValue() API and passing it the handles of the monitored keys. The monitoring threads then fall into an infinite waiting state with WaitForSingleObject() until the change notification event is triggered.

For example, as soon as any subkey/value of the following registry key is altered Look2Me will restore it immediately:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

DEAD LOOP

To summarize the privilege attack description, let's have a look at the Figure 1 below and define the removal issues associated with this technique.

In order to detect and delete the adware file, the user-mode process needs winlogon.exe to be started without the Look2Me module loaded as its extension. This can be achieved if the Look2Me registry entries are removed and the system is restarted.

However, the Look2Me registry entries cannot be removed because of the spawned monitoring threads. As soon as these entries are altered or deleted, Look2Me restores them immediately.

To prevent registry entries from being recreated, the Look2Me threads need to be terminated or suspended.

The problem with this is that no user-mode process will have the privilege to call OpenProcess() with a powerful access mode (e.g. PROCESS_ALL_ACCESS, PROCESS_TERMINATE) to manipulate the Look2Me process/threads. In this case, OpenProcess() will fail and the GetLastError() will return an 'Access is denied' error.

To manipulate processes and threads, SeDebugPrivilege must first be restored.

A user-mode process may allocate and initialize SID for the BUILTINS/Administrators and then enumerate its rights. If it detects that the Administrator has no SeDebugPrivilege enabled, it may grant this privilege with the LsaAddAccountRights() API.

The next thing the user-mode process will need to do is to adjust its own SeDebugPrivilege to the SE_PRIVILEGE_ENABLED attribute. However, in order for this privilege to be truly enabled in the access token of the BUILTINS/Administrators account (so that it can be inherited by the Administrator processes), the system must be rebooted.

As the system reboot is invoked, the Logoff system event notification will call the Look2Me Winlogoff() API. Then, with a new logon session, winlogon.exe will start up, load Look2Me, and call its exported WinLogon() API again, notifying it about the Logon system event. Both times Look2Me fires up and strips SeDebugPrivilege from all accounts again, so that BUILTINS/Administrators will not

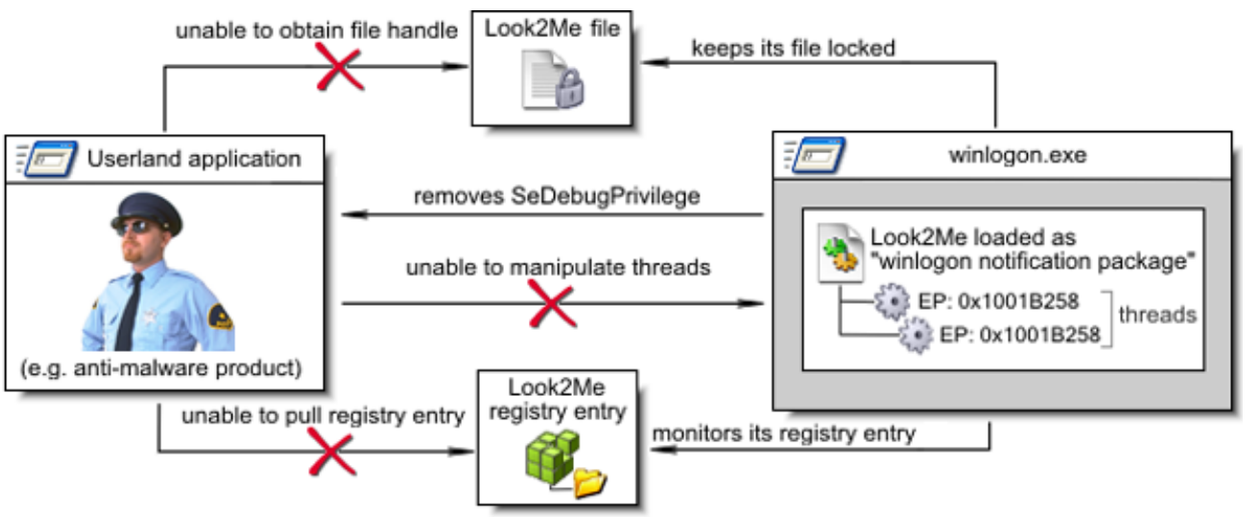


Figure 1.

have this privilege enabled in the newly created logon session. This leads to the dead loop.

RISING FROM THE ASHES

Let's consider what can still be done in this situation.

Our user-mode application is still capable of dropping its own component and installing it as 'winlogon notification package'. For this purpose, the application may register the exported APIs to handle the system events Logon, Startup, and StartShell. The Asynchronous value may need to be set to one to have its APIs called by winlogon.exe in the separate threads.

Next, our process should reboot the system. After the reboot, the system events Logon, Startup, and StartShell will be spawned in separate threads within our winlogon extension. A race condition with Look2Me might be expected in this case. Every thread would need to wait until the privilege is found to be stripped, then restore it, and quit.

The privileges defined at this stage will shape the token of the authenticated user and they will be inherited by other processes.

The user-mode process should then be able to enumerate running processes. For every process it will be able to enumerate its threads, read them, and detect the Look2Me threads by signature.

Once the entry points of the detected adware threads and the address ranges of the loaded modules are known, it is easy to find out within which modules the threads were spawned. As the Look2Me modules are identified inside every scanned process, their fullpath filenames will need to be collected for further reference.

In addition to this, every detected Look2Me thread needs to be terminated or suspended (either will work). Thread deactivation will block the automatic recreation of the adware registry entries.

As a result of the fact that the CLSID of Look2Me and the registry entries are random, the user-mode process needs to enumerate all registry keys and their values under the following registry keys:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

To locate the keys that should be removed, the values '(default)' and 'DllName' need to be inspected in the following registry keys (respectively) to determine whether they are set to any of the module fullpath filenames that were collected in the previous step:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\[Enumerated CLSID]\InprocServer32
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\[Enumerated Subkey]

The located keys belong to Look2Me and they must be deleted.

In addition, the CLSID of Look2Me can also be collected to remove the values from the registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Shell Extensions\Approved

'PRESS OK TO CRASH YOUR SYSTEM NOW'

If our user-mode application acts to reboot or power off, this will invoke Logoff and Shutdown system events, which in turn, will spawn new Look2Me threads inside the modules that are still loaded. This will repeat the whole payload again.

How can the system be shut down with no Logoff and Shutdown system events triggered? ExitWindowsEx() with EWX_POWEROFF and EWX_FORCE is not a cure – it still invokes the events mentioned.

One solution would be to patch the exports of the Look2Me module within the winlogon.exe process. A simpler method would be to 'power off' the machine by terminating winlogon.exe itself. The system crash and the subsequent reboot will run winlogon.exe with no Look2Me loaded (assuming the registry was cleaned properly) so that system scan can successfully be started again to clean the remaining Look2Me files.

CONCLUSION

The method described here allows the successful removal of Look2Me in user mode. Nevertheless, this piece of adware provides food for thought about the restrictions of user mode and indicates that the next meeting point with 'unwanted' software may take place in the kernel mode – and it seems only to be a question of time until that happens.

[Jason Bruce, of SophosLabs, will present a paper on defining the rules for 'acceptable' adware at this year's Virus Bulletin conference. VB2005 takes place 5–7 October 2005 in Dublin, Ireland. The abstract for Jason's paper, as well as the full conference programme and online registration can be found at http://www.virusbtncom/conference/.]

TECHNICAL FEATURE

PROBLEMS IN STATIC BINARY ANALYSIS – PART 2

Aleksander Czarnowski
AVET Information and Network Security, Poland

In the first part of this article (see *VB*, May 2005 p.12) we inspected several problems that are encountered when particular objects are loaded into memory. In this part we will inspect further problems associated with static analysis techniques. We assume that the object and accompanying libraries have been loaded successfully, and that we have our first disassembly ready.

THE FIRST LOOK ...

In the case of obfuscated code the first disassembly is usually far from perfect – even when using advanced tools such as *IDA Pro* or *OllyDBG* that analyse the code before providing the user with a code disassembly output. Figures 1 and 2 demonstrate a very simple code obfuscation technique based on prefixing instructions with segment registers and/or REP/REPNE opcodes. Inspection of the

```
DATA:00DE2000 60                pusha
DATA:00DE2001 E8 00 00 00 00    call    $+5
DATA:00DE2006
DATA:00DE2006                   loc_DE2006:
DATA:00DE2006 5D                pop     ebp
DATA:00DE2007 8B C5             mov     eax, ebp
DATA:00DE2009 83 E8 06          sub     eax, 6
DATA:00DE200C 81 ED 06 20 DE 00 sub     ebp, offset loc_DE2006
DATA:00DE2012 60                pusha
DATA:00DE2013 8A C3             mov     al, bl
DATA:00DE2015                   db      65h, 36h
DATA:00DE2015 65 36 F3 8B CA    rep mov ecx, edx
DATA:00DE201A F2 EB 01          repne jmp short loc_DE201E
DATA:00DE201A                   ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
DATA:00DE201D AB                db 0ABh ; ź
```

Figure 1: Use of GS:, REP: and REPNE: prefixes to obfuscate code during disassembly; this output was produced by IDA Pro after initial analysis.

```
00DE2000  60              PUSHAD
00DE2001  E8 00000000     CALL 0x90.00DE2006
00DE2006  5D              POP EBP
00DE2007  8BC5            MOV EAX,EBP
00DE2009  83E8 06         SUB EAX,6
00DE200C  81ED 0620DE00   SUB EBP,0x90.00DE2006
00DE2012  60              PUSHAD
00DE2013  8AC3            MOV AL,BL
00DE2015  65:             PREFIX GS:
00DE2016  36:F3:          PREFIX REP:
00DE2018  8BCA            MOV ECX,EDX
00DE201A  F2:             PREFIX REPNE:
00DE201B  EB 01           JMP SHORT 0x90.00DE201E
00DE201D  AB              STOS DWORD PTR ES:[EDI]
```

Figure 2: Use of GS:, REP: and REPNE: prefixes to obfuscate code during disassembly; this output was produced by OllyDBG during the debugging session of the same code.

DE201A address reveals an interesting code structure: a jump opcode prefixed with REPNE opcode. The use of REP/REPNE prefixes can pose problems for static analysis tools: it is not possible in every case to guess the CPU state that would influence further program execution.

Take a look at the DE2015 address in Figures 1 and 2 – *IDA* failed to disassemble this byte stream fully, while *OllyDBG* decided that the 65h opcode is the GS: prefix and disassembled the whole stream. This brings us to an important observation: different tools can disassemble the same code differently. In the real world things are a bit more complicated as most tools use different mnemonics for disassembly, which makes data exchange and data correlation even harder.

Why is this so important? Simply: if we create a tool that uses the disassembly listing as its input, then we probably cannot stop the disassembly process straight after the initial analysis. We first need to 'clean' the disassembly output. This is true both for normal compiler-generated code and for obfuscated code.

ALWAYS JUMP IN THE MIDDLE

Another interesting case that readers can play with is the challenge described in [1]. At the time of writing this article the solution to the challenge has not been published. I don't want to spoil the fun, so we will look only at the beginning of the file. When loading this object into the *IDA* disassembler, there is a warning that should ring alarm bells (see Figure 3).

Loading this file with *OllyDBG* provides us with another indication that the entry point is outside the code sections – the debugger issues this warning during file loading. Take a look at this snippet of the disassembly generated by *IDA*:

```
seg002:00407BD6 E9 25 E4 FF FF  jmp   loc_406000
[…]
seg002:00406000                       loc_406000:
seg002:00406000 60              pusha
seg002:00406001 F8              clc
seg002:00406002 E8 02 00 00 00  call  near  ptr
                                      loc_406007+2
seg002:00406007                       loc_406007:
seg002:00406007 E8 00 E8 00 00  call  near  ptr  41480Ch
seg002:0040600C                 db    0
seg002:0040600D                 db    0
seg002:0040600E                 db    5Eh ; ^
[…]
```

If we were to feed a static analysis tool with this disassembly it would generate the wrong results. The reason is the CALL instruction at the 406002 address. While *IDA* calculated the procedure address correctly (406007 + 2), it did not influence the code disassembly. If we count

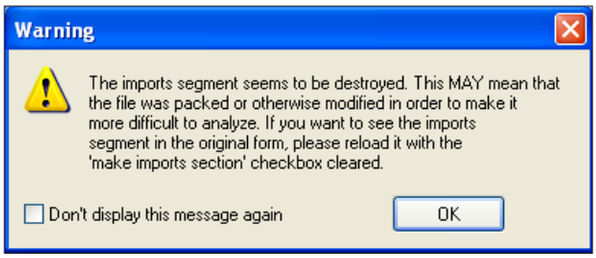


Figure 3: IDA detects 'corruption' of the import segment while loading challenge.exe.

instruction bytes it is obvious that the first CALL instruction is calling procedures that start in the middle of another CALL at the 406007 address.

Now let's use *IDA's* interactive functionality to correct this disassembly. As a quick fix I converted the bytes at loc_406007 to data ('d' key), moved the cursor to the correct address (406007 + 2 =) 406009 and converted the bytes from that address to code ('c' key). (Note that this is not the correct method of fixing such disassembly problems in *IDA*. You should add cross-reference instead of just converting bytes to code.) Here is the result:

```
seg002:00406000                         loc_406000:
seg002:00406000 60              pusha
seg002:00406001 F8              clc
seg002:00406002 E8 02 00 00 00  call loc_406009
seg002:00406007 E8              db 0E8h    ;junk code
seg002:00406008 00              db    0
seg002:00406009                         loc_406009:
seg002:00406009 E8 00 00 00 00  call $+5
seg002:0040600E 5E              pop  esi
seg002:0040600F 2B C9           sub  ecx,  ecx
seg002:00406011 58              pop  eax
seg002:00406012 74 02           jz   short near ptr
                                     loc_406014+2
seg002:00406014
seg002:00406014                         loc_406014:
seg002:00406014 CD 20 B9 51 19 00 VxDCall 1951B9h
seg002:0040601A 00 8B C1 F8 73 02 add [ebx+273F8C1h], cl
seg002:00406020 CD 20 83 C6 33 8D VxDJmp 8D334683h
```

At address 406012 we see a trick that is similar to the one described before, but this time instead of using CALL the author of this code used the JZ instruction. Also take a look at the 406014 address: the two bytes (CD20 = INT 20) that are skipped over were chosen wisely to make the disassembler think this is a VxD call. Of course it's the B9 51 19 00 bytes that really counts:

```
seg002:00406012 74 02           jz  short loc_406016
seg002:00406014 CD              db  0CDh  ;INT 20 opcode
seg002:00406015 20              db  20h   ;junk code
seg002:00406016                           loc_406016:
seg002:00406016 B9 51 19 00 00  mov ecx,  1951h
```

The two code obfuscation techniques presented above are enough to demonstrate a whole set of problems associated with static analysis tools that use disassembly as their input. We need to do a lot of work on the disassembly listing before feeding it into another tool for further analysis.

Does this mean that we should disregard static analysis methods? Absolutely not. After all, we should remember the advantages of this approach, which include not needing to run code (and create processes and threads) and the ability to analyse code for different CPU architecture and operating systems.

Now it's time to solve our problems – at least partially.

CODE EMULATION TO THE RESCUE

We can strengthen our static analysis and disassembly process by adding full code emulation. This allows us to gain some advantages that previously were reserved for dynamic analysis tools like debugger. *IDA* seems to be a good target – after all it is a very powerful disassembler, which provides plug-in functionality through its SDK (note: *IDA* SDK is virtually undocumented, so your best bet is to analyse somebody else's plug-in code).

A perfect example of such an approach is the ida-x86emu plug-in by Chris Eagle [2]. In [3] there is a discussion of how this tool was used successfully against UPX, Burneye and Shiva for example. Another reason to use ida-x86emu is the fact that this is an open source project, making this an excellent starting point for extending it.

It is worth noting that ida-x86emu not only works successfully against some code obfuscation techniques, but can also help in bypassing dynamic analysis protection. A good example is the use of the RDTSC instruction to measure execution time for a particular code snippet. If the code is single-stepped execution, time increases enormously and this is easy to detect. However, ida-x86emu emulates the RDTSC instruction and internal counters – take a look at its source code:

```
case 0x30: //
      if (opcode == 0x31) { //RDTSC
          edx = (dword) (tsc >> 32);
          eax = (dword) tsc;
      }
```

The tsc value is increased after every opcode emulation.

EXTENDING IDA-X86EMU – A SHORT CRASH COURSE

While working with ida-x86emu I decided that it would be convenient to have a current emulated line displayed in the

x86emu window just like register values and stack. This proved to be a nice exercise in understanding how *IDA* internals really work. Because ida-x86emu emulates the CPU it is not really interested in the line number but in the current position in terms of code. This is kept in ea_t (line address of instruction). The loc variable of type ea_t is initialized according to the eip variable, which reflects the EIP register value.

To obtain the filled cmd structure which holds the internal instruction representation (*IDA* internal representation of the instruction is different from the instruction opcode value) I used the ua_ana0() function. To get the disassembly line that *IDA* generates I needed two more functions:

- generate_disasm_line(eip, opstr, sizeof(opstr),0); – generates one line of disassembly from code at eip location
- tag_remove(opstr, opstr, sizeof(opstr)); – removes additional tags from disassembly line so it can be easily displayed in static text control

The rest of the modifications are simple Win32 API functions used to display the text in the plug-in window.

METAPROCESSOR

While extending the ida-x86emu plug-in I also wanted to be able at a later point to use some kind of metaprocessor over the disassembled code. I could, of course, save the results to a text file after running the emulator over it. However, all the pieces of the metaprocessor are already in this plug-in and I wanted to show how easy it could be to write one using existing tools.

Before we delve further into technical aspects I should define the metaprocessor term. In our case the metaprocessor is not working on real CPU instruction – instead it works on an abstract view of emulated/disassembled code. This allows us to work only on relevant code sections like the analysis of flow control. A similar technique is used, for example, in a binary comparison based on graphs [4]. An important feature of the metaprocessor is the fact that the same metaprocessor can be used for different CPU architectures. The only difference is a code that translates real opcode sets into abstract instructions.

A very simple metaprocessor is presented in [5]. The approach described in [5] is interesting as the whole solution has been developed in Perl and is based on objdump for providing input disassembly. A similar simple tool developed in Python with the help of dumpbin is demonstrated in [6] as a proof of concept. In fact, Perl, Ruby and Python are very well suited as environments for developing metaprocessors based on externally generated disassembly in text format.

Extending ida-x86emu in order to perform additional analysis with an external metaprocessor developed in one of these scripting languages is a fairly trivial task. For simplicity I decided to use the *IDA* output window. The msg() function from SDK allows us to output the string in this window. Later the metaprocessor can be fed with the result from the *IDA* output window. To make parsing of the result from the output window easier it is a good idea to add some prefixes (such as inst:) before the metaprocessor instruction.

SUMMARY

The object of this two-part series was to present different obfuscation and anti-analysis techniques and illustrate their impact on the static analysis of binary objects. While we worked on *Windows* PE files, most of the techniques could be used in the Unix world as well. The difference lies in the executable file format (ELF or A.OUT) and system loader internals and the fact that Unix/*Linux* systems lack great tools such as *OllyDBG* and *SoftICE* to mention just a couple.

It seems that static analysis backed up by metaprocessor, graph analysis and code emulation is a very powerful combination technique, which can greatly automate the disassembly of obfuscated code. Further development of these methods will allow not only better malware analysis but also vulnerability detection in binary objects and powerful binary comparison.

ACKNOWLEDGEMENTS

I would like to thank Chris Eagle for answering my questions about ida-x86emu.

BIBLIOGRAPHY

[1] http://www.t2.fi/english/challenge-05.html.

[2] ida-x86emu: http://ida-x86emu.sourceforge.net/.

[3] Chris Eagle, ‘idax86emu x86 Emulator Plugin for IDA Pro’, CODECon 2004 http://ida-x86emu.sourceforge.net/codecon04.pdf.

[4] Halvar Flake, ‘Structural Comparison of Executable Objects’, http://www.sabre-security.com/files/dimva_paper2.pdf.

[5] Cyrus Peikari, Anton Chuvakin, *Security Warrior*, O’Reilly, January 2004, ISBN 0-596-00545-8; http://www.oreilly.com/catalog/swarrior/.

[6] Aleksander Czarnowski, ‘Win32 API in Binary Application Security Analysis’, Cyprus InfoSec 2004.

COMPARATIVE REVIEW

WINDOWS XP

Matt Ham

VB's last comparative review on *Windows XP* (see *VB*, June 2004, p.12) was carried out at around the same time as the release of *XP Service Pack 2*. Fortunately for the products, the release date of *SP2* was just after the deadline for the comparative, thus the products were spared the challenge of having to perform on the newly updated platform. Having had close to a year in which the products could adapt to the new features in *SP2*, this month's review was expected to bring few surprises and not to be too taxing.

Happily, this was indeed the case. The testing process was the smoothest that I can remember, with only a handful of crashes to mar the plain sailing. Considering the instability problems I usually encounter on other platforms this is convincing evidence that *Windows XP* bears the bulk of testing, whether this be by developers in-house, or at the hands of end users. All but one of the products on offer integrated fully with the Windows Security interface, which was a slightly higher percentage than I had expected.

However, there were problems in two other areas. Of more immediate importance to users, there was a significant upsurge in the number of false positives generated while scanning the clean sets. This meant that a VB 100% award was denied to more than one of the products in the review. On a more personal level, the logging attempts by some products ranged from the downright disgraceful to the perplexingly cryptic.

THE TEST SETS

The test sets were aligned to the February 2005 WildList, with a product submission deadline of 3 May 2005. This time lag should have been enough for all but the most tardy developers to catch up with detection, thus high detection rates were expected. The additions to the In the Wild (ItW) test set were a dull bunch, as ever, and possibly the most uninspiring yet. The predominance of various W32/*bot samples does not give cause for further comment.

AhnLab V3 Pro 2004 6.0.0.383

ItW Overall	100.00%	**Macro**	99.00%
ItW Overall (o/a)	100.00%	**Standard**	97.72%
ItW File	100.00%	**Polymorphic**	74.82%

Testing of *V3 Pro* this month progressed with few problems, and detection rates proved perfect for ItW viruses. There were no other problems that were relevant to VB 100% status, thus *AhnLab* is in receipt of the award this month.

However, problems were encountered during on-access testing of *V3Pro*. Somewhat unusually, the 'leave as is' option for on-access detection does not deny access to infected files. Thus infected files were deleted instead of logging denied access attempts. *V3Pro* is also unusual in that it does not scan archives by default. The option was activated when scanning archives during the clean set timings.

Alwil avast! antivirus 4.6.654[vps 0518-1]

ItW Overall	100.00%	**Macro**	99.56%
ItW Overall (o/a)	100.00%	**Standard**	99.12%
ItW File	100.00%	**Polymorphic**	93.58%

On-access scanning with *avast!* started problematically, with an error proclaiming that ashEnhcd was out of memory. Scanning also seemed very slow. As has been noted in previous reviews, this was due to the fact that all viruses detected on access are added to the quarantine area, even when the quarantine option is not activated. In this case it seemed that the resultant filling of the OS partition also denied the system virtual memory, hence the error.

The timing function within the product was also rather eccentric. Since these timers are often flawed, external timing is used for the clean set scans and then compared against the product's listed timings. In the case of *avast!* it seems that the internal timer starts not from zero, but from five seconds, thus adding considerable illusory overhead to fast scans.

Aside from these oddities, *avast!* performed admirably on other fronts, and obtained a VB 100% award easily.

ArcaBit ArcaVir 2005

ItW Overall	99.96%	**Macro**	98.99%
ItW Overall (o/a)	99.96%	**Standard**	99.22%
ItW File	99.96%	**Polymorphic**	85.90%

ArcaVir was the odd man out in this test as far as stability was concerned, with the on-access scanner crashing repeatedly after 300 or so infected files had been thrown at it. To circumvent this problem the tests were performed with the scanner set to delete infected files, and repeated until no further infections were logged.

The product was troubled in other areas too, with AntiCMOS.A missed on access in the boot set and the .HTM form of W32/Nimda.A In the Wild. That Nimda can

cause problems so long after its release is an enduring mystery to me. A false positive in the clean test sets completed *ArcaVir*'s woes, with this adding to the miss of the ItW Nimda sample to deny the product a VB 100%.

Authentium Command AntiVirus 4.92.91

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.72%
ItW File	100.00%	**Polymorphic**	99.95%

Command AntiVirus is a long-standing entrant in *VB*'s testing regime and thus it comes as no surprise that problems with the product were few and far between. However, there were a number of issues with the log file which caused some grief. First, the log file is available only as an RTF file, which increases its size appreciably. This might not be such a problem if the log were not truncated before export can occur, since a more compact log would be expected to be truncated less, if at all. Due to these logging problems the on-demand tests were performed by deleting infected files and examining those left. While logging was problematic the other aspects of testing were not, with a VB 100% award being the result.

Avira Avira 1.00.00.64

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

With most products in this test the installation procedure either mentions a reboot explicitly or ignores the issue entirely. In the case of *Avira* a reboot is deemed to be recommended, but not vital – which makes it a little unclear as to what might be changed by the reboot process. *VB*'s test procedures include a reboot in any case. Detection rates have improved once more for *Avira*, and are now very good, with no misses either on access or on demand. With no false positive detections either, the result is a VB 100% award for *Avira*.

BLC Win Cleaner 7.03

ItW Overall	100.00%	**Macro**	98.03%
ItW Overall (o/a)	100.00%	**Standard**	96.39%
ItW File	100.00%	**Polymorphic**	96.43%

The detection rates of *Win Cleaner* and its parent product *Quick Heal* remain high, though the results in this test were rather overshadowed by the issue of false positives. In total 28 false positives were generated during clean set scanning – certainly enough to give cause for concern and equally sufficient for a VB 100% to be denied. Of some note was the presence among these false positives of a detection for 'Hoax.Pompol'.

These two products also share the dubious distinction of being the last to present log file entries in a strict 8+3 format, a feature which complicates parsing of the logs no end.

CA eTrust Antivirus (I) 7.1.192

ItW Overall	100.00%	**Macro**	99.90%
ItW Overall (o/a)	100.00%	**Standard**	99.82%
ItW File	100.00%	**Polymorphic**	99.89%

CA's *eTrust Antivirus* supports two engines, this being an optional setting with the InocuLAN engine activated. Updating was particularly seamless, to the extent that I assumed it must have failed due to being so fast and not interrupting the on-access scanner. As ever, all is well with the product until the log files are encountered. These are so outrageously poor that the designer should be chained to a rock and his liver devoured by eagles in the ancient fashion. Not only do the results for single files stretch over several lines due to word wrapping, but the word wrapping is continued over several columns – fragmenting the results beyond any ease of parsing, either automatically or by observation.

CA eTrust Antivirus (Vet) 7.1.192

ItW Overall	100.00%	**Macro**	99.88%
ItW Overall (o/a)	100.00%	**Standard**	99.70%
ItW File	100.00%	**Polymorphic**	99.87%

With the same interface as the preceding product, this is the version with the default setting as far as the engine is concerned, and is thus eligible for a VB 100% award. Since the scanning results were good and no false positives arrived to spoil the proceedings, a VB 100% award is awarded. The logging was, however, the same abomination as with the alternative engine.

CA Vet Anti-Virus 10.66.0 11.8.00

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.72%
ItW File	100.00%	**Polymorphic**	99.87%

On-access tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
AhnLab V3 Pro	0	100.00%	0	100.00%	100.00%	47	98.97%	3187	74.80%	70	96.16%
Alwil avast!	0	100.00%	0	100.00%	100.00%	18	99.56%	112	93.58%	19	98.93%
ArcaBit ArcaVir 2005	1	99.96%	1	100.00%	99.96%	34	99.45%	1308	85.97%	22	98.91%
Authentium Command	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.95%	5	99.58%
Avira Avira	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
BLC Win Cleaner	0	100.00%	0	100.00%	100.00%	86	97.96%	339	96.43%	474	72.45%
CA eTrust Antivirus (I)	0	100.00%	0	100.00%	100.00%	4	99.90%	1	99.89%	4	99.51%
CA eTrust Antivirus (V)	0	100.00%	0	100.00%	100.00%	12	99.82%	2	99.87%	5	99.60%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	2	99.87%	6	99.54%
CAT Quick Heal	0	100.00%	0	100.00%	100.00%	86	97.96%	339	96.43%	474	72.45%
Doctor Web Dr.Web	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.69%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	2	99.63%
Fortinet FortiClient	1	99.96%	0	100.00%	99.96%	660	84.19%	122	94.83%	62	97.46%
FRISK F-Prot Antivirus	0	100.00%	0	100.00%	100.00%	0	100.00%	6	99.97%	8	99.40%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.85%
GDATA AntiVirusKit	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Grisoft AVG	0	100.00%	0	100.00%	100.00%	3	99.93%	757	83.64%	34	98.17%
H+BEDV AntiVir	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Hauri ViRobot	0	100.00%	3	0.00%	99.50%	0	100.00%	49	98.83%	18	98.96%
Kaspersky KAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	2	99.88%
McAfee VirusScan	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.79%
MicroWorld eScan	0	100.00%	0	100.00%	100.00%	9	99.71%	0	100.00%	0	100.00%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	2	99.95%	181	91.03%	8	99.50%
NWI Virus Chaser	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.69%
SOFTWIN BitDefender	0	100.00%	0	100.00%	100.00%	0	100.00%	6	99.73%	14	99.33%
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	8	99.80%	0	100.00%	14	99.33%
Symantec SAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
UNA UNA	12	97.57%	0	100.00%	97.58%	1891	55.06%	14264	20.28%	489	77.45%
VirusBuster VirusBuster	0	100.00%	0	100.00%	100.00%	0	100.00%	108	92.58%	18	98.98%

Another of those products where there is nothing to be said but words of praise, *Vet AntiVirus* is destined for a short write-up confirming that its performance was worthy of a VB 100% award. *Vet* remains unique in that an out-of-date version of the product refuses to scan, forcing the user either to update or have no scanning functionality at all. Quite how effective this is with real users – who are not always known for choosing security over convenience – is a matter for conjecture.

CAT Quick Heal 7.03

ItW Overall	100.00%	**Macro**	98.03%
ItW Overall (o/a)	100.00%	**Standard**	96.39%
ItW File	100.00%	**Polymorphic**	96.43%

Quick Heal is identical in all but appearance to its daughter product *BLC Win Cleaner*. As such the comments made for that product are directly applicable for *Quick Heal*. Sadly for *CAT*, this includes the withholding of a VB 100% award due to the generation of 28 false positives in the clean test set.

Doctor Web Dr.Web 4.32b

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

Dr.Web remains admirable in every way other than the configuration of its on-access scanner. This requires a reboot after any configuration change, including such matters as changing the default log size, which might be classified as relatively minor. The tray icon for the scanner also vanished at one point, seemingly a configuration change triggered merely by opening a dialog rather than actually changing settings.

However, this is minor stuff in comparison with the detection rates shown by *Dr.Web*, which gains yet another VB 100%.

Eset NOD32 1.1087

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.63%
ItW File	100.00%	**Polymorphic**	100.00%

The results for *NOD32* were, once more, somewhat perplexing for a product which claims not to scan within archives. Despite this claim it detected samples of W32/Heidi.A in their zipped form, suggesting that such scanning may be activated by default.

On this occasion *Eset*'s scanner missed two samples in the standard set, though this was not sufficient to deny the company another VB 100%.

Fortinet FortiClient 2.27 8.812

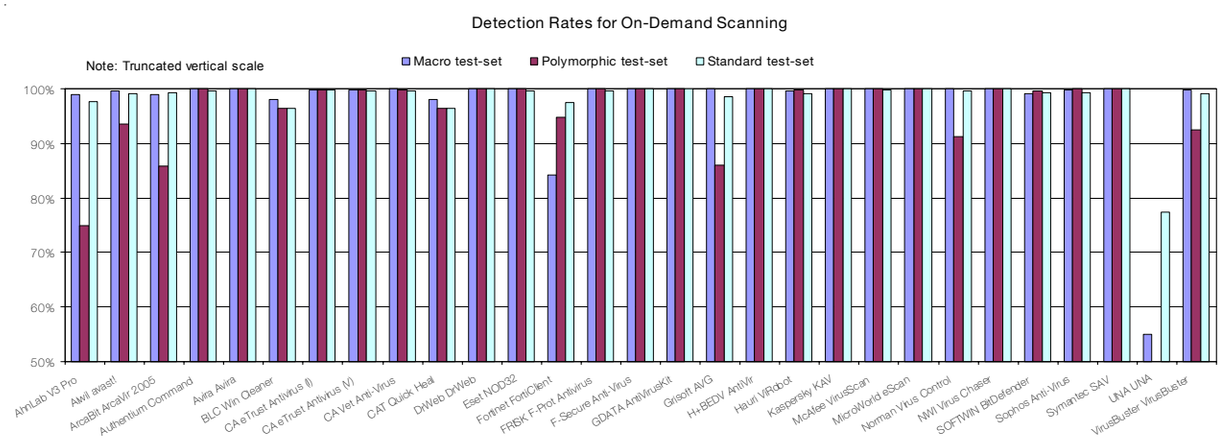
ItW Overall	99.96%	**Macro**	84.19%
ItW Overall (o/a)	99.96%	**Standard**	97.41%
ItW File	99.96%	**Polymorphic**	94.72%

FortiClient's VB 100% aspirations were not to be realised this month after the product became another victim of ancient viruses and, like *ArcaVir*, was unable to detect the .HTM form of W32/Nimda.A In the Wild. Other than this (admittedly rather major) flaw, *FortiClient*'s performance was good.

FRISK F-Prot Antivirus 3.16b

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.56%
ItW File	100.00%	**Polymorphic**	100.00%

During normal testing *FRISK*'s submission for this test demonstrated no problems whatsoever, the result of which is a VB 100% award for *F-Prot*. However, an error on my part highlighted an odd feature of the product. As a matter of routine, on-access scanners are deactivated during testing of



On-demand tests	ItW File		ItW Boot		ItW Overall	Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	%	Number missed	%	Number missed	%	Number missed	%
AhnLab V3 Pro	0	100.00%	0	100.00%	100.00%	52	99.00%	3180	74.82%	62	97.72%
Alwil avast!	0	100.00%	0	100.00%	100.00%	18	99.56%	112	93.58%	17	99.12%
ArcaBit ArcaVir 2005	1	99.96%	0	100.00%	99.96%	70	98.99%	1312	85.90%	19	99.22%
Authentium Command	0	100.00%	0	100.00%	100.00%	0	100.00%	1	99.95%	2	99.72%
Avira Avira	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
BLC Win Cleaner	0	100.00%	0	100.00%	100.00%	81	98.03%	340	96.43%	101	96.39%
CA eTrust Antivirus (I)	0	100.00%	0	100.00%	100.00%	4	99.90%	1	99.89%	1	99.82%
CA eTrust Antivirus (V)	0	100.00%	0	100.00%	100.00%	10	99.88%	2	99.87%	4	99.70%
CA Vet Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	2	99.87%	3	99.72%
CAT Quick Heal	0	100.00%	0	100.00%	100.00%	81	98.03%	340	96.43%	101	96.39%
Doctor Web Dr.Web	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Eset NOD32	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	2	99.63%
Fortinet FortiClient	1	99.96%	0	100.00%	99.96%	660	84.19%	123	94.72%	63	97.41%
FRISK F-Prot Antivirus	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	6	99.56%
F-Secure Anti-Virus	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	2	99.92%
GDATA AntiVirusKit	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Grisoft AVG	0	100.00%	0	100.00%	100.00%	0	100.00%	257	85.97%	27	98.56%
H+BEDV AntiVir	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Hauri ViRobot	0	100.00%	3	0.00%	99.50%	12	99.71%	9	99.76%	16	99.08%
Kaspersky KAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
McAfee VirusScan	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	3	99.79%
MicroWorld eScan	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
Norman Virus Control	0	100.00%	0	100.00%	100.00%	2	99.95%	180	91.24%	6	99.63%
NWI Virus Chaser	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
SOFTWIN BitDefender	0	100.00%	0	100.00%	100.00%	35	99.10%	6	99.73%	14	99.33%
Sophos Anti-Virus	0	100.00%	0	100.00%	100.00%	8	99.80%	0	100.00%	15	99.30%
Symantec SAV	0	100.00%	0	100.00%	100.00%	0	100.00%	0	100.00%	0	100.00%
UNA UNA	12	97.57%	0	100.00%	97.58%	1891	55.06%	14264	20.28%	489	77.45%
VirusBuster VirusBuster	0	100.00%	0	100.00%	100.00%	15	99.81%	109	92.55%	17	99.17%

on-demand functionality. This should make no difference in theory, as one would expect that a scanner would be instructed not to scan on access a file which it is opening to scan on demand. In practice, however, this is not always the case. When the *F-Prot* on-access scanner was inadvertently left running during an on-demand test the result was to show several files that had been blocked by the on-access scanner. This behaviour has been observed in other products in the past, but usually goes unnoticed due to the testing methodology.

F-Secure Anti-Virus Client Security 5.55 SR1

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.92%
ItW File	100.00%	**Polymorphic**	100.00%

F-Secure Anti-Virus is very much in a state of predictability these days, with all but full detection in the test sets. Only W32/Heidi.A concealed within zipped files and W32/Nimda.A in its TMP file form were missed. Since both samples require a degree of interaction to turn into an infectious object, such misses can hardly be considered a problem. Part of the predictable nature of *FSAV* is its string of VB 100% awards, to which it adds another on this occasion.

GDATA AntiVirusKit 15.0.5

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

Continuing in its successful ways, *AVK* once more detected all files in the test set both on access and on demand. Clearly, the combination of engines used by *AVK* is capable of good protection, though speed issues might be a problem for some impatient users. A VB 100% award is duly winging its virtual way to *GDATA*.

Grisoft AVG Anti-Virus 7.0.308

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	98.56%
ItW File	100.00%	**Polymorphic**	85.97%

Another product which causes no problems and produces no surprises is *Grisoft*'s *AVG*, which earns another VB 100% award. Only one feature was irritating enough for me to note, which was that the timings for scans are not kept on screen after the scan has completed. With this the most serious problem encountered, it can be appreciated that the product is not brimming with faults.

H+BEDV AntiVir 6.30.00.18

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

The results for *AntiVir* are identical to those for sister product *Avira*. The only differences noted were cosmetic, with the graphics in *Avira* being noticeably more up to date than those sported by *AntiVir*. The two products' similarities extend to the award of a VB 100% for their excellent performance.

Hauri ViRobot Desktop 5.0 149168

ItW Overall	99.50%	**Macro**	99.71%
ItW Overall (o/a)	99.50%	**Standard**	99.08%
ItW File	100.00%	**Polymorphic**	99.76%

Testing of *ViRobot* started well but was beset by a number of problems later in the process. The clean test set saw the first problems, with a number of false positives emerging. Intriguingly, one of these was a detection of the confusingly named 'Not-A-Virus.15718'. There was also a total inability to detect the floppy disk-based samples in the test sets.

Logging proved annoying, since exporting to file seemed not to work at first. The export did complete eventually, though this was after an interval of several minutes had passed, with the testing process having passed on to other matters by this time. On-access scanning was also tricky, with most of the usual avenues used in these tests blocked. In the end the scanner was set to disinfect and CRC testing was used to determine which files had been affected. Despite good detection rates in the file-based sets, the false positive detections and the missed floppy detections mean that *ViRobot* is denied a VB 100% award in this test.

Kaspersky KAV Personal 5.0.227

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

KAV has now settled back into its traditional pattern of repeated VB 100% awards after suffering a brief glitch a few months ago. While perfect detection across all test sets will be gratifying to *Kaspersky*, it leaves me little to say, other than to reveal that I was wearing *Kaspersky* socks while performing the tests.

McAfee VirusScan Enterprise 8.0.0 4400 4483

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	99.79%
ItW File	100.00%	**Polymorphic**	100.00%

Hard Disk Scan Rate	Executables			OLE Files			Zipped Executables		Zipped OLE Files	
	Time (s)	Throughput (kB/s)	FPs [susp]	Time(s)	Throughput (kB/s)	FPs [susp]	Time (s)	Throughput (kB/s)	Time(s)	Throughput (kB/s)
AhnLab V3 Pro	31.0	17643.0		7	11333.4		90	1771.3	18	4144.9
Alwil avast!	122.0	4483.1		9	8814.9		54	2952.2	23	3243.8
ArcaBit ArcaVir 2005	216.0	2532.1	1	7	11333.4		65	2452.6	8	9325.9
Authentium Command	96.0	5697.2		4	19833.4		49	3253.4	5	14921.5
Avira Avira	470.0	1163.7		4	19833.4		189	843.5	16	4663.0
BLC Win Cleaner	78.0	7012.0	28	15	5288.9		57	2796.8	21	3552.7
CA eTrust Antivirus(I)	150.0	3646.2		4	19833.4		57	2796.8	9	8289.7
CA eTrust Antivirus (V)	160.0	3418.3		5	15866.8		65	2452.6	11	6782.5
CA Vet Anti-Virus	147.0	3720.6		7	11333.4		70	2277.4	12	6217.3
CAT Quick Heal	80.0	6836.7	28	17	4666.7		55	2898.5	20	3730.4
Doctor Web Dr.Web	139.0	3934.8		9	8814.9		62	2571.2	12	6217.3
Eset NOD32	99.0	5524.6		11	7212.2		40	3985.4	9	8289.7
Fortinet FortiClient	325.0	1682.9		9	8814.9		54	2952.2	23	3243.8
FRISK F-Prot Antivirus	166.0	3294.8		5	15866.8		79	2017.9	9	8289.7
F-Secure Anti-Virus	189.0	2893.8		23	3449.3		102	1562.9	25	2984.3
GDATA AntiVirusKit	460.0	1189.0		16	4958.4		199	801.1	20	3730.4
Grisoft AVG	163.0	3355.4		7	11333.4		66	2415.4	9	8289.7
H+BEDV AntiVir	424.0	1289.9		4	19833.4		173	921.5	12	6217.3
Hauri ViRobot	488.0	1120.8	3 [1]	34	2333.3		311	512.6	53	1407.7
Kaspersky KAV	112.0	4883.3		13	6102.6		69	2310.4	17	4388.7
McAfee VirusScan	69.0	7926.6		11	7212.2		65	2452.6	15	4973.8
MicroWorld eScan	370.0	1478.2		32	2479.2		148	1077.1	62	1203.3
Norman Virus Control	594.0	920.8		6	13222.3		239	667.0	6	12434.6
NWI Virus Chaser	136.0	4021.6		10	7933.4		59	2702.0	11	6782.5
SOFTWIN BitDefender	507.0	1078.8		7	11333.4		150	1062.8	8	9325.9
Sophos Anti-Virus	97.0	5638.5		12	6611.1		74	2154.3	19	3926.7
Symantec SAV	173.0	3161.5		21	3777.8		62	2571.2	20	3730.4
UNA UNA	69.0	7926.6	14	98	809.5		9	17713.0	22	3391.2
VirusBuster VirusBuster	202.0	2707.6	[1]	29	2735.6		118	1351.0	30	2486.9

VirusScan showed few problems in detection rates during these tests, though there were some noteworthy irritations with the interface. Since choosing an area to scan requires both selecting dropdown menus and browsing, the process is tedious to perform on multiple occasions. *McAfee* has opted not to scan archives by default, and this lack of archive scanning was responsible for two of the three misses observed (detection of archived versions of W32/Heidi.A being impossible without handling the zip files in which they are located). These quibbles aside, good detection rates and the lack of false positives earn *VirusScan* a VB 100%.

MicroWorld eScan Internet Security 2.6.522.9

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

As a rebadged version of *GDATA*'s *AVK*, the detection rates for *eScan* were expected to be very good. It was a little strange, however, to see that *eScan* missed samples of three W97M viruses which presented *AVK* with no problems at all.

There were also initial problems with the interface, with the 'leave alone' option on detection seeming to have no effect. This proved only to be momentary, however. With no other problems *eScan* gained a VB 100% award despite being somewhat enigmatic.

Norman Virus Control 5.80 5.82.01

ItW Overall	100.00%	**Macro**	99.95%
ItW Overall (o/a)	100.00%	**Standard**	99.63%
ItW File	100.00%	**Polymorphic**	91.24%

With the complexity of its sandbox emulation engine running, the slow speeds of scanning for *Norman*'s product are an expected, if irritating, feature. The technology does not detect all the more complex polymorphics in the test set but the detection for *NVC* elsewhere is good. No false positives were detected (in fact none have been seen for the product in living memory), meaning that *Norman* earns its latest VB 100% award.

NWI Virus Chaser 5.0

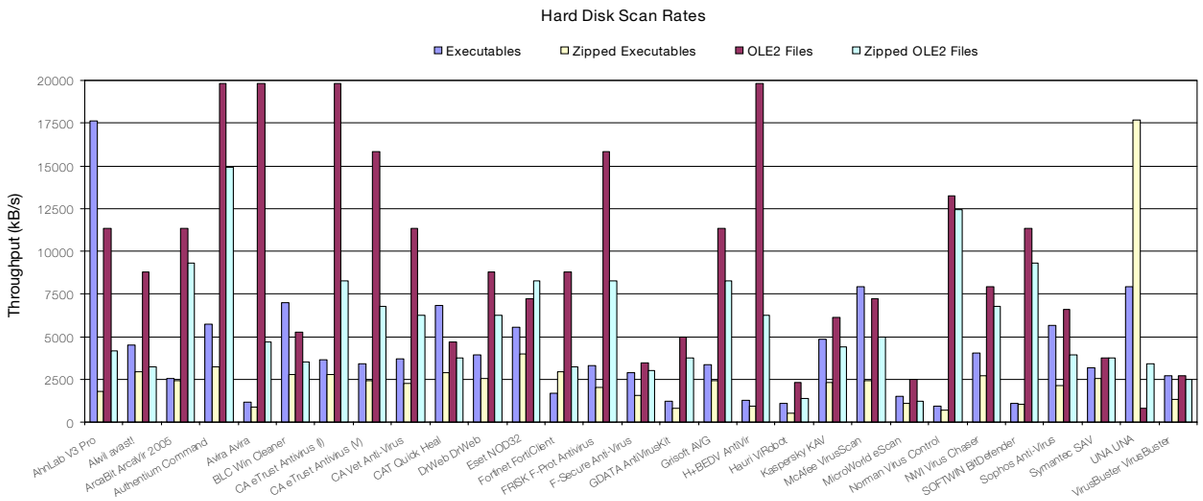
ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

Virus Chaser is a rebadged version of *Dr.Web*, the similarities between the products being obvious from installation onwards. The similarities include the requirement for a reboot between changes to settings and the need to activate the on-access scanner after installation. However, the installation routine does not mention this at all – and no reboot is prompted after reconfiguration, leaving the user never quite sure of the current settings. The log too is rather less than desirable, splitting file names from paths and thus complicating the use of the results. Despite these problems, detection is very good and no false positives were generated, thus *Virus Chaser* earns a VB 100% award.

SOFTWIN BitDefender 8 Professional Plus 7.01144

ItW Overall	100.00%	**Macro**	99.10%
ItW Overall (o/a)	100.00%	**Standard**	99.33%
ItW File	100.00%	**Polymorphic**	99.73%

There was just one area where problems occurred with *BitDefender*. During on-access testing, the 'deny access and continue' option produced file errors rather than simply 'access denied' as a result of attempting to access the files. This problem also occurred with the use of Xcopy, even when the



'ignore errors' switch was used. As a result, deletion was chosen as the setting, noting the files which were not deleted. Even here there were problems, since some files were disinfected rather than deleted. Therefore CRC checking was also used to determine which files were, in fact, not detected. These problems should not impact a normal user seriously, however, and do not prevent the award of a VB 100% to *BitDefender*.

Sophos Anti-Virus 5.0.1

ItW Overall	100.00%	**Macro**	99.80%
ItW Overall (o/a)	100.00%	**Standard**	99.30%
ItW File	100.00%	**Polymorphic**	100.00%

Sophos Anti-Virus has undergone something of an image change in the latest release, which might not be altogether a good thing. Rather than being ugly to look at, but pleasant to use, the product is now pleasant to look at, but ugly to use. The default location of the log file has also changed to deep within the Documents tree. This may be aligned with *Windows* file location good practice but it always enrages me when searching for logs. The shock of the new aside, *Sophos Anti-Virus* remains its usual self as far as detection is concerned, and gains a VB 100% as reward.

Symantec SAV 9.0.0.338 51.1.0.15

ItW Overall	100.00%	**Macro**	100.00%
ItW Overall (o/a)	100.00%	**Standard**	100.00%
ItW File	100.00%	**Polymorphic**	100.00%

Symantec SAV demonstrated a very good detection rate with no false positives, thus earning another VB 100% award. However, the results were marred by the logging facilities within the program. The whole application crashed during the creation of the log file, though the file itself was created correctly. The log file is viewable in several different places in the GUI, but among these views there seems an arbitrary division as to where exporting of the results can be performed. More disturbing is the treatment of some viruses in the log file. As has been noted before, seeing a reference to a virus discovered in file '????????' is not particularly useful.

UNA UNA 1.83 265

ItW Overall	97.58%	**Macro**	55.06%
ItW Overall (o/a)	97.58%	**Standard**	77.45%
ItW File	97.57%	**Polymorphic**	20.28%

UNA distinguished itself on this occasion with the dubious honour of being the only product not to be recognised by Windows Security Center. This resulted in rather more irritating pop-up bubbles than usual while testing proceeded. It also managed 14 false positives in the clean test set, mostly claiming the presence of HLLO.NumberOne.K and HLLP.Jacklyn.12416. There were also issues with the on-access scanner. Although no reboot is required after installation, the on-access scanner does not seem to perform until the machine is rebooted. Detection rates continue to improve, but there is still some way to go before a VB 100% award is achieved by *UNA*.

VirusBuster VirusBuster Professional 2005 5.0.163

ItW Overall	100.00%	**Macro**	99.81%
ItW Overall (o/a)	100.00%	**Standard**	99.17%
ItW File	100.00%	**Polymorphic**	92.55%

The last product in the line up is another which does not scan archives by default. This became more noticeable during scanning of the clean test sets, when several configuration switches were required. These changes seemed to be very sluggish, with an irritating delay between updating the settings and the GUI updating itself. These are not major problems, however, and detection rate was good. One file was detected as suspicious in the clean sets but this was not a full blown false positive, thus a VB 100 % can be awarded.

CONCLUSION

As mentioned in the introduction, this was one of the least problematic of the recent reviews from a technical point of view. Log files remain an issue which seems destined never to vanish, though numerous companies have changed their logs for the better. The increase in false positives is something of a worrying trend. Whether it will turn out to be a momentary blip, or increase in future tests remains to be seen.

Technical details:

Test environment: Identical 1.6 GHz Intel Pentium machines with 512 MB RAM, 20 GB dual hard disks, DVD/CD-Rom and 3.5-inch floppy drive running Windows XP Professional SP2.

Virus test sets: Complete listings of the test sets used can be found at http://www.virusbtn.com/Comparatives/WinXP/2005/test_sets.html.

A complete description of the results calculation protocol can be found at http://www.virusbtn.com/Comparatives/Win95/199801/protocol.html.

END NOTES & NEWS

The 3rd annual BCS IT Security Conference takes place on 7 June 2005 in Birmingham, UK. The conference focuses on identity theft, hacking, cyber-terrorism, network forensics, secure web services, encryption and related topics. See http://www.bcsinfosec.com/.

NetSec 2005 will be held 13–15 June 2005 in Scottsdale AZ, USA. The programme covers a broad array of topics, including awareness, privacy, policies, wireless security, VPNs, remote access, Internet security and more. See http://www.gocsi.com/events/netsec.jhtml.

The 17th annual FIRST Conference will be held 26 June – 1 July 2005 in Singapore. The conference provides a forum for sharing goals, ideas, and information on how to improve global computer security. The five-day event comprises two days of tutorials and three days of technical sessions. For more information see http://www.first.org/conference/2005/.

A SRUTI 2005 workshop entitled 'Steps to Reducing Unwanted Traffic on the Internet' takes place 7–8 July 2005 in Cambridge, MA, USA. The workshop aims to bring academic and industrial research communities together with those who face the problems at the operational level. See http://www.research.att.com/~bala/sruti/.

Black Hat USA takes place 23–28 July 2005 in Las Vegas, NV, USA. Training will take place 23–26 July and the Briefings will take place 27–28 July. For details and online registration see http://www.blackhat.com/.

The 14th USENIX Security Symposium will be held 1–5 August 2005 in Baltimore, MD, USA. For more information see http://www.usenix.org/.

T2'05, the second annual T2 conference, will be held 15–16 September 2005 in Helsinki, Finland. The conference focuses on newly emerging information security research. All presentations are technically oriented, practical and include demonstrations. See http://www.t2.fi/english/.

The Network Security Conference takes place 19–21 September 2005 in Las Vegas, NV, USA. The conference is designed to meet the education and training needs of the seasoned IS professional as well as the newcomer. For details see http://www.isaca.org/.

The 5th Annual FinSec Conference takes place 20–23 September 2005 in London, UK. This year's conference will focus on the unique set of challenges afflicting information security professionals in the financial community. See http://www.mistieurope.com/.

The 15th Virus Bulletin International Conference, VB2005, will take place 5–7 October 2005 in Dublin, Ireland. The programme for the three-day conference can be found on the *VB* website. For more information or to register online see http://www.virusbtn.com/.

Black Hat Japan (Briefings only) will be held 17–18 October 2005. Further details will be announced at the Black Hat USA event in July. See http://www.blackhat.com/.

RSA Europe 2005 will be held 17–19 October 2005 in Vienna, Austria. For more details see http://www.rsaconference.com/.

WORM 2005 (the 3rd Workshop on Rapid Malcode) will take place 11 November 2005 in Fairfax, VA, USA. The workshop will provide a forum to bring together ideas, understanding and experiences bearing on the worm problem from a wide range of communities, including academia, industry and the government. The organisers are currently seeking submissions from those wishing to present at the workshop. Full details can be found at http://www1.cs.columbia.edu/~angelos/worm05/.

The eighth Association of Anti-Virus Asia Researchers International Conference (AVAR 2005), takes place in Tianjin, China on 17 and 18 November 2005. The theme of this year's conference will be 'Wired to Wireless, Hacker to Cybercriminal'. The organizers are currently seeking submissions from those wishing to present at the conference, the deadline for submissions is 10 June 2005. For more details see http://aavar.org/

Infosecurity USA will be held 6–8 December 2005 in New York, NY, USA. The conference will take place 6–8 December, with the accompanying exhibition running 7–8 December. For more details see http://www.infosecurityevent.com/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery: £195 (US$358)

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

NEWS & EVENTS

CHANGING CAN-SPAM

The US Federal Trade Commission (FTC) has invited public comment on a number of changes it proposes making to the CAN-SPAM Act.

One of the Commission's proposals is to shorten the deadline by which senders must honour recipients' opt-out requests. Currently, senders have 10 business days to comply with opt-out requests, but the FTC proposes shortening this to just three business days. However, the Direct Marketing Association (DMA) has expressed concern about this proposal, saying its members feel that such a timeframe would be 'unworkable'. In addition, the FTC proposes to make it illegal for email senders to require recipients to pay a fee or supply information in order to unsubscribe.

Other changes to the Act include clarifying the definitions of 'sender' and 'person' to help identify a single sender for each message, and including Post Office boxes in the definition of 'valid physical postal address'. Comments on the proposals (which can be found at http://www.ftc.gov/os/2005/05/05canspamregformfrn.pdf) are required by 27 June 2005.

EVENTS

Inbox/Outbox takes place on 22 and 23 June 2005 in London, UK. The event will cover every aspect of inbound and outbound email. See http://inbox-outbox.com/.

CEAS 2005, the Second Conference on Email and Anti-Spam, will be held 21–22 July 2005 at Stanford University, CA, USA. For more details see http://www.ceas.cc/.

TREC 2005, the Text Retrieval Conference, will be held 15–18 November 2005 at NIST in Gaithersburg, MD, USA. For more details see http://trec.nist.gov/.

FEATURE

BAYESIAN CLASSIFICATION APPLIED TO PHISHING FRAUD

Eugene Koontz, Jonathan Oliver, Andrew Klein and Christine Drake
MailFrontier, Inc., USA

'Phishing' is the term for an email scam that spoofs legitimate companies in an attempt to defraud the recipient of personal information such as logins, passwords, credit card numbers, bank account information and social security numbers. For example, an email that appears to come from *PayPal* may claim that the recipient's account information must be verified because it may have been compromised by a third party. However, when the recipient provides the account information for verification, the information is sent to a fraudster, who is then able to access the recipient's account. The term phishing was coined because the fraudsters are 'fishing' for personal information.

Phishing emails are sent both to consumers and companies, attempting to gain either personal information from an individual or confidential information about an enterprise. In phishing email messages, the sender must gain the trust of the recipients to convince them to divulge information. The fraudsters attempt to gain credibility through mimicking or 'spoofing' a legitimate company through methods such as using the same logos and corporate colour scheme, changing the 'from' field so that the email appears to come from someone in the spoofed company, and adding some legitimate links in the email. Figure 1 shows an example of a phishing email.

Once credibility has been established, the fraudulent email will present a plausible premise that requests the recipient to act. For example, the email may claim that the recipient's account information is outdated, a credit card has expired, or that the account has been selected randomly for verification. The request is framed in an urgent situation requiring a quick response. There are numerous approaches and each tries to create a scenario that would convince the recipient that they must provide the requested information.

To collect the information, the fraudulent email generally provides either a form in the email or a link to a fraudulent website. The submit button on the form sends the

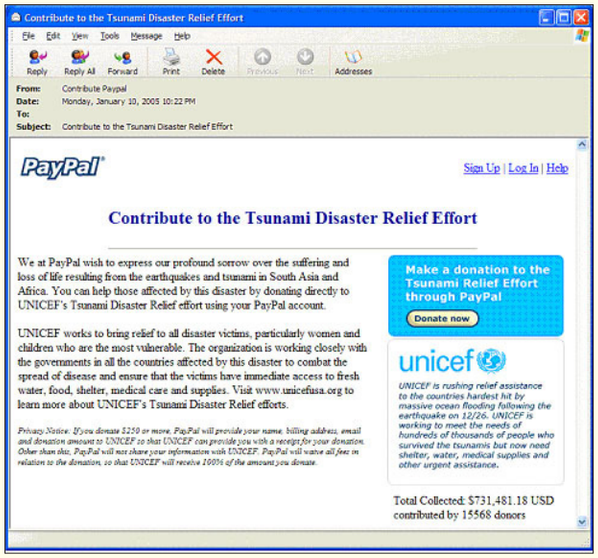


Figure 1. Consumer fraud spoofs PayPal and asks recipients to donate to the Tsunami Relief Effort. Users' personal information is stolen when they attempt to make a donation.

information to the fraudsters. Fraudulent web pages have fields for information which is sent to the fraudsters. Once the information has been gathered it can be used for identity theft or financial gain, or to access critical company information.

IDENTIFYING PHISHING EMAILS

Not only must phishing emails be caught, but they must also be categorized specifically as fraudulent emails. They cannot just be grouped in email junkboxes along with spam. Phishing emails can do substantial damage and recipients are easily fooled because they are designed to look like good email. For example, financial institutions are spoofed more often than organisations from any other industry. People expect to receive correspondence from their banks and are likely to believe that the phishing emails are valid requests for information.

MailFrontier has posted a Phishing IQ Test on its website, which tests whether people can correctly identify phishing emails (http://survey.mailfrontier.com/survey/quiztest.html). The test has been taken by over 300,000 people. The results show that the participants misidentified the emails 28 per cent of the time; fraudulent emails were misidentified as legitimate at a rate of 31 per cent and legitimate emails were misidentified as fraudulent at a rate of 19 per cent [1].

Phishing emails need to be classified as fraud to prevent users from believing that the emails are legitimate correspondence that were accidentally labelled as spam. *MailFrontier* found that, when phishing emails were placed in a general junkbox, they were 'unjunked', or moved back to the inbox, up to ten per cent of the time. Spam emails, in contrast, are unjunked at a rate which is significantly less than 0.25 per cent.

Although it is important that phishing emails be labelled as fraud, it is crucial to avoid any false positives, which misidentify good emails as fraud. Users will expect the fraudulent emails to resemble legitimate correspondence and will most likely not be able to identify when good emails, possibly requiring important transactions, have been mislabelled as fraud.

An effective fraud filter will target features that are specific to phishing emails, identifying these emails as fraud while letting good emails pass through.

APPLYING BAYESIAN FILTERS TO PHISHING

Bayesian filtering has a well established history as an anti-spam weapon [2–5]. However, Bayesian spam filters are ineffective at identifying phishing emails. Spam emails are generally sales pitches aimed at promoting a product or service. Phishing emails, on the other hand, are designed to look like legitimate transactional correspondence and almost always work to disguise their true source. To catch phishing emails accurately, Bayesian filters must be designed specifically to identify the unique aspects of phishing emails.

Evaluating text

Naïve Bayesian filtering can be applied to identifying phishing emails. The probability that an email is fraud, *P(F|E)*, can be calculated using Bayes' Rule:

$$P(F \mid E) = \frac{P(E \mid F) * P(F)}{P(E)}$$

A similar calculation can be made for non-fraud. These calculations include the unknown, but fixed, value *P(E)*. This value can be ignored because it is simply a scaling factor and only relative values, and not absolute values, are needed. Thus the equation would be reduced to the following:

$$P(F \mid E) \propto P(E \mid F) * P(F)$$

Each email is considered to be a set of words and features $W_0 \ldots W_n$. To determine *P(E|F)* in the calculation above, the product of the conditional probabilities for each feature W_i must be calculated. Naïve Bayes assumes that each feature appears independently and the probability that an

email is fraud is simply the product of the probabilities for each feature.

$$P(E|F) = P(W_0|F) * P(W_1|F) * \ldots * P(W_n|F) = \Pi_i P(W_i|F)$$

Inserting this into the calculation for *P(F|E)*:

$$P(F|E) \propto \Pi_i P(W_i|F) * P(F)$$

Some phishing emails contain language that would not be used in legitimate transactional correspondence, making it easy to catch with simple Naïve Bayesian filtering applied to the words in the email. However, the majority of phishing emails are sophisticated enough that they will not be identified as fraud merely based on an analysis of the email text. Other features specific to phishing must be included as part of the analysis.

Including fraud indicators

When applying Bayesian filtering to phishing, various phishing indicators must be included in addition to text analysis to generate an effective result. Some indicators of phishing include:

- Links based on raw IP addresses instead of domain names.
- '@' symbols used in URLs.
- Null characters in hostnames.
- Illegal or double-redirects of URLs.
- Other methods of obscuring and encoding URLs.
- Inconsistent contact points (URLs and phone numbers within the email are from different sources).
- Non-standard ports (using ports other than 80).
- Similar, but illegitimate domain names (for example, paypalverify.net instead of paypal.com).

The value of a fraud feature in discriminating fraud from non-fraud can be determined by examining the odds ratio for that feature:

$$Odds\ Ratio = \frac{P(W_i \mid F)}{P(W_i \mid NF)}$$

where NF is non-fraud.

Based on the examples shown in Table 1, the presence of the word 'verify' in the subject of an email is the strongest indicator of fraud. When the odds ratios of different tokens are considered together, the probability that a particular email is fraud can be estimated effectively. Because text is not a strong indicator of fraud, these more intelligent features need to be layered as part of the Bayesian fraud analysis.

Token	Odds Ratio
SUBJ_verify (the word 'verify' appears in the subject)	368.7
PHRASE_verify_your_identity (the phrase 'verify your identity' is found in the email)	206.44
SUBJ_debit (the word 'debit' appears in the subject)	165.7
SUBJ_bank (the word 'bank' appears in the subject)	159.2
URL_:4903 (URL is directed to port 4903)	194.4
WORD_suspension (the word 'suspension' is found in the email)	142.59
URL_cit (URL contains 'cit' for Citibank fraud)	139.8
URL_:87 (URL is directed to port 87)	139.8
LINK_INCONSISTENT (text of the HTML link differs from the actual target)	50.40

Table 1. Examples of features and their odds ratio showing the odds that the feature will appear in fraud over non-fraud emails.

Training from datasets

Once the features have been selected, the Bayesian fraud filter must be trained to determine the probability that a particular occurrence of that feature is an indicator of fraud.

For example, if non-standard ports are selected as a feature, the Bayesian fraud filter must be trained to determine the probability that a specific non-standard port (e.g. port 5880) is an indicator of fraud.

To train a Bayesian spam filter, only two datasets are required: a dataset of spam and a dataset of good email. However, in order to apply Bayesian filtering to fraud, two additional datasets are needed: legitimate transactional email and phishing email. A large set of legitimate transactional email is needed because this set of email most resembles phishing emails and the filter must have numerous examples of legitimate transactional email to help ensure a low false positive rate.

Applying the filter

After the filter has been trained on sufficiently large datasets, it can be applied to incoming emails. First, Bayesian detection methods are applied, which causes three groups to emerge (see Figure 2): good email, spam email, and transactional email (including both legitimate and fraudulent transactional email).

Once the transactional emails have been separated, the intelligent fraud features are applied to make a fraud

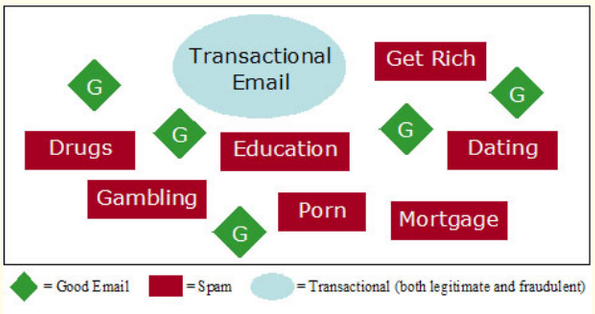


Figure 2. First step in fraud filtering: applying Bayesian detection methods to develop a set of transactional emails.

judgment. This will separate the fraud emails from the legitimate transactional emails.

Testing the filter

MailFrontier trained a Bayesian fraud filter using the methods discussed above and then tested the filter for effectiveness. The test emails included the following sets:

- 2,193 fraudulent emails
- 1,177 legitimate transactional emails
- 12,978 good emails not related to transactions

These emails were passed through the Bayesian fraud filter and gave the following results.

	Fraud set	**Legitimate transactional set**	**Good set**
Set Count	2,193	1,177	12,978
False Negative	582 (27%)		
False Positive		0 (0%)	2 (0%)

Table 2. Bayesian test results.

The methods used were set to guarantee a very low false positive rate; zero per cent of the legitimate transactional emails were misidentified as fraud. As a result of ensuring a low false positive rate, some fraudulent emails were not identified as fraud; there was a 27 per cent false negative rate. The overall result was 73 per cent of fraudulent emails were caught, while all legitimate transactional emails were delivered, showing that Bayesian methods identify phishing fraud successfully.

When the Bayesian fraud filter was applied, other email filtering methods were not used to ensure a true test of the filter's effectiveness. Methods that were not used included:

- Reputation services
- Authentication
- Whitelisting
- Real-time black lists
- Real-time phishing link lists

When these methods are added to Bayesian fraud filtering, the effectiveness of fraud detection increases further while maintaining the low false positive rate.

CONCLUSION

Effective email security solutions must protect users from all types of email threats. However, each threat must be treated differently to address its unique properties. Bayesian filtering has been successful in combating spam. However, this same filter cannot be applied to fraud. Instead, a Bayesian fraud filter can be created through layering different features which are strong indicators of phishing.

Test results show that Bayesian rules can effectively be applied to identifying fraud. This identification allows phishing emails to be categorized uniquely. Even if recipients cannot recognize the emails as fraud, the emails can be removed securely from the inbox and labelled appropriately, keeping users safe.

This article is based on a presentation given at the 2005 Spam Conference at MIT in January 2005. Authors of the presentation are Eugene Koontz, Jon Oliver, and Andrew Klein. The presentation PowerPoint can be found at http://www.mailfrontier.com/docs/mit_2005_BayesFraud.pdf.

REFERENCES

[1] Drake, Christine; Klein, Andrew; and Oliver, Jonathan; 'Analysis of the phishing email problem and discussion of possible solutions', presented at the Third International Workshop on Security in Information Systems, May 2005.

[2] Graham, Paul; 'A plan for spam', August 2002, http://paulgraham.com/spam.html.

[3] Graham-Cumming, John; 'Naïve Bayesian text classification', *Dr. Dobb's Journal*, May 2005. Retrieved from http://www.ddj.com/documents/s=9698/ddj0505a/0505a.html.

[4] Pantel, Patrick and Dekang, Lin, 'SpamCop: a spam classification & organization program', poster in *Proceedings of AAAI-1998 Workshop on Learning for Text Categorization*, July 1998.

[5] Sahami, Mehran, *et al.,* 'A Bayesian approach to filtering junk e-mail', *Proceedings of AAAI-1998 Workshop on Learning for Text Categorization*, July 1998.