MAY 2004

virus BULLETIN

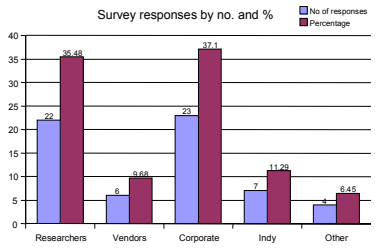
The International Publication on Computer Virus Prevention, Recognition and Removal

## CONTENTS



ISSN 0956-9979

## IN THIS ISSUE




### THE ELUSIVE UNICORN

The results of Andrew Lee's malware naming survey indicate that the anti-virus industry's UNICORN is as elusive as the shy mythical creature itself.

**page 13**

### CHIBA WITTY BLUES

W32/Witty shows a number of similarities to W32/Slammer: it is short, its sending rate is limited only by available bandwidth, and it selects random target IP addresses. Unlike Slammer, however, this worm features a very destructive payload. Frédéric Perriot, Péter Ször and Peter Ferrie provide the witty comments.

**page 9**

### vbSpam supplement

This month: anti-spam news and events; closing loopholes in the mail flow with MTA Mark and SPF; and a summary of the month's posting to the ASRG mailing list.

*"When it comes to malware protection, overconfidence is our own worst enemy."*

**Larz Sherer**
**Independent researcher, USA**

## KEEPING PACE IN A WAR OF WORMS

Malware authorship is a never-ending battle of one-upmanship. Worm A cascades quickly across the Internet, only to be superceded by worm B, followed by worm A version 2. After a successful worm rampages across the Internet, the next one may attempt to remove the previous one – it's a chance to claim a share of the spotlight and derail one's rivals in the same stroke. The current case in point is Mydoom being upended by Netsky and Bagle.

A few years ago, experts predicted that the use of polymorphic and metamorphic qualities in malware would increase, to make them more evasive. This has not been the case. The complexities involved in the construction of such code make it more or less non-palatable to malware authors. Though there are some polymorphic aspects to worms, on the whole, worms do not adapt on the fly.

The surprising pattern seen today is that malware behaviour is becoming less complex. The de facto standard for worms is to contain a garden-variety SMTP engine, have the ability to harvest email addresses from the hard drive, attempt to terminate active processes of known AV and/or security software and spread through open network shares. A few new tricks can be expected but by and large, the techniques are strikingly redundant. While some worms attempt to do too much and end up doing very little except being a nuisance through propagation techniques, their method of gaining access to computers in the first place is usually the tried and true strategy of social engineering. Even for more technically savvy users with AV software, firewall appliances and router protection in place, there is a prevailing assumption that internal network addressing schemes are inherently shielded from attack because machines are not using a public IP address. When it comes to malware protection, overconfidence is our own worst enemy. Proving this theory is the effective infiltration technique of SIN – Static IP Notification.

Most remote administration tools function with the client (in the hands of the attacker) being able to connect to an infected machine by way of a server application (a Trojan) on the victim computer. The concept of SIN is to reverse this so that the infected server application connects to the client. For this to work, the attacker must have a static IP address. This puts the attacker at risk of discovery, so the IP address is masked by using a chain of anonymous web hosting accounts, configured with customized CGI scripts for DNS redirection along with a proxy server, fake domain or all three working in tandem. What makes this effective is the fact that the infected server application uses trusted traffic such as HTTP and DNS to access the Internet. In some cases, that traffic is encrypted, making egress filtering on the router or firewall even less effective.

Trojans have evolved to a stage where near-complete stealth is possible, but few individuals possess the skill to use them at expert level. It has been a tireless exercise in futility for malware authors to package Trojan features and tools within a worm only to discover it does not work correctly. Worms that are configured to contact defined email addresses, domains or websites can easily be combated by terminating those destinations, preventing the worm from unleashing its intended potential. The use of SIN could result in a reversal of fortune for malware authors, since it uses pathways that are not easily defended in outbound DNS and HTTP. A number of elegantly crafted Trojan horses make SIN a potent threat and it is only a matter of time before SIN, along with other, evolving means of endpoint infiltration, becomes recognized as a means of making corporate networks fair game in malware author one-upmanship.

In protecting against malware, from simple to complex, the most effective approach is to remain vigilant. Utilize technologies that enable flexibility between rigid policy control and the empowerment of users as well as administrators to make educated decisions based on experience. Ready machines in your environment for a defensive posture at all times.

# NEWS

## NAI REVERTS TO MCAFEE

In a bid to reduce costs and increase productivity, *Network Associates, Inc.* (*NAI*) has announced plans to streamline its business and to change its name to *McAfee, Inc.* The company intends to focus solely on providing security products and services, and in order to do so, is selling its network and application performance management business, *Sniffer*, to *Silver Lake Partners* and *Texas Pacific Group*. The purchasers, who are expected to pay $275 million for the business, intend to form a standalone company named *Network General Corporation*, which will continue to sell the *Sniffer* products. Upon completion of the sale, *Network Associates* will adopt its (not so) new name, *McAfee, Inc.*

The complexities of name changes and corporate re-branding can often seem a mystery to the end-user. In this instance, the company is (more or less) reverting to its old name: *McAfee Associates* merged with *Network General* back in December 1997 to form *Network Associates Inc.* (see *VB*, November 1997, p.3). While the company itself got a new name, the *McAfee* product tag has remained in the public eye since 1989, making it an obvious choice for the company's latest incarnation.

## SYMANTEC VULNERABILITIES

*eEye Digital Security* has reported that it has discovered four new vulnerabilities affecting *Symantec* products. Products affected are *Norton Internet Security 2004*, *Norton Internet Security 2004 Professional* and *Norton Personal Firewall 2004*. *eEye* rates the severity of all four vulnerabilities as 'high' and describes three of them as 'remotely-exploitable vulnerabilit[ies] that allow anonymous attackers to compromise default installations of the affected software and gain absolute access to the host machine' and the fourth as 'a remotely-exploitable vulnerability that allows an anonymous attacker to execute a severe denial-of-service attack against systems running default installations of the affected software.' Another *Symantec* vulnerability is currently awaiting the release of a patch. *eEye* employs a policy of releasing only minimal details of vulnerabilities until the manufacturer of the software concerned has released a patch. Nevertheless, March 2004 saw a buffer overflow vulnerability employed by a worm only 24 hours after its publication by *eEye* – see p.9.

## CALL FOR PAPERS: AVAR 2004

AVAR (the Association of anti-Virus Asia Researchers) has issued a call for papers for AVAR 2004 in Tokyo, which will take place 25–26 November 2004 in Tokyo, Japan. More details can be found at http://www.aavar.org/.

### Prevalence Table – March 2004

| Virus | Type | Incidents | Reports |
|---|---|---|---|
| Win32/Netsky | File | 232,716 | 89.09% |
| Win32/Bagle | File | 19,234 | 7.36% |
| Win32/Dumaru | File | 4,698 | 1.80% |
| Win32/Klez | File | 743 | 0.28% |
| Win32/Sobig | File | 728 | 0.28% |
| Win32/Mydoom | File | 722 | 0.28% |
| Win32/Swen | File | 577 | 0.22% |
| Win32/Bugbear | File | 319 | 0.12% |
| Win32/Mimail | File | 195 | 0.07% |
| Win32/Nachi | File | 145 | 0.06% |
| Win32/Sober | File | 99 | 0.04% |
| Win32/Fizzer | File | 79 | 0.03% |
| Redlof | Script | 71 | 0.03% |
| Win32/SirCam | File | 67 | 0.03% |
| Win32/Gaobot | File | 56 | 0.02% |
| Win32/Valla | File | 53 | 0.02% |
| WYX | Boot | 52 | 0.02% |
| Win32/Gibe | File | 51 | 0.02% |
| Win32/Lovsan | File | 43 | 0.02% |
| Win32/Funlove | File | 40 | 0.02% |
| Win95/Spaces | File | 40 | 0.02% |
| Win32/Opaserv | File | 38 | 0.01% |
| Win32/Magistr | File | 35 | 0.01% |
| Win32/Nimda | File | 31 | 0.01% |
| Win32/Elkern | File | 24 | 0.01% |
| Win32/Hybris | File | 24 | 0.01% |
| Win32/Yaha | File | 21 | 0.01% |
| Laroux | Macro | 19 | 0.01% |
| Psyme | Script | 18 | 0.01% |
| Fortnight | Script | 16 | 0.01% |
| Win32/Lovgate | File | 16 | 0.01% |
| Win32/Sdbot | File | 16 | 0.01% |
| Others[1] | | 231 | 0.09% |
| Total | | 261,217 | 100% |

[1]The Prevalence Table includes a total of 231 reports across 72 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

# VIRUS ANALYSIS 1

## NETSKY: CONFLICT STARTER?

*Mircea Ciubotariu*
BitDefender, Romania

*As VB goes to print, the latest version of W32/Netsky doing the rounds is Netsky.Z. Here, Mircea Ciubotariu looks at variants .A to .K and their impact on the AV world.*

Whatever the motivations for creating and spreading malware – interest, revenge, curiosity, or anything else – it seems to be evolving into a real industry. It has two characteristics of concern: innovation and competition. Unfortunately competition both drives improvements and increased efficiency and feeds the needs of survival and ego.

### FROM THE DEEP

W32/Netsky is a typical worm which spreads primarily through mail and secondarily via P2P sharing applications. However, the P2P spreading function exists only in the .A, .B and .C variants.

P2P spreading is achieved by the worm copying itself in all directories containing the string 'share' or 'sharing' on the local system and on mapped network drives – in the .C variant the author discovered it was simpler only to look for the string 'shar'. The .A and .B variants copy themselves using 26 enumerated file names, while the .C variant copies itself using 44 file names. A full listing of these filenames can be found at http://www.virusbtn.com/netsky.xmltab#1.

The worm's file format is Windows 32-bit executable, compiled with Visual C++ 6 and packed with a number of different packers (see table below). All versions up to .K are compiled for release. The later .L and .M variants are compiled for debug.

In addition to the regular time and date attributes found in any ordinary file, Windows 32-bit executables have within their PE header a dword field called timestamp. This encodes the local date and time upon generation of the .exe file and is located at offset +08h (relative to the PE header). This field is not altered by most of the packers and both permits analysis of the worm's propagation and, more importantly, keeps track of the release of different variants. This is very helpful in the case of malware with many versions, such as Netsky.

| Variant | Timestamp | Packer | Size |
|---|---|---|---|
| Netsky.A | 13/02/04, 15:50 | UPX 1.24 | 21,504 bytes |
| Netsky.B | 16/02/04, 18:48 | UPX 1.24 | 22,016 bytes |
| Netsky.C | 24/02/04, 14:51 | UPX 1.24 | 24,064 bytes |
| | 24/02/04, 17:22 | ASPack | 28,160 bytes |
| | 24/02/04, 21:37 | Petite | 25,352 bytes |
| Netsky.D | 29/02/04, 09:59 | Petite | 17,424 bytes |
| Netsky.E | 27/02/04, 15:49 | Petite | 24,840 bytes |
| Netsky.F | 02/03/04, 21:50 | PEPack | 18,432 bytes |
| Netsky.G | 03/03/04, 21:07 | PEPack | 27,648 bytes |
| Netsky.H | 03/03/04, 14:07 | PEPack | 22,528 bytes |
| Netsky.I | 03/03/04, 18:21 | PEPack | 22,016 bytes |
| Netsky.J | 05/03/04, 23:41 | tElock | 27,648 bytes |
| Netsky.K | 01/03/04, 16:09 | tElock | 22,016 bytes |

Making the assumption that each variant was released a short while after it was built, we observe that the .E variant was released before variant .D; variants .H and .I were released before variant .G, and variant .K was released before variant .F.

The trick of changing packers between variants does not present a problem for AV products as most have already implemented unpackers to go through the packed (and sometimes encrypted) data to access the original code.

### CRAWLING ON THE GROUND

When Netsky runs it checks a mutex to avoid re-infection. If it finds the mutex it quits. Because this infection marker differs among the variants, several versions of the worm should be able to run simultaneously on the same system; however, this is not always possible because the later variants disable earlier ones (as shall be explained later). The mutexes are:

| | |
|---|---|
| Netsky.A | AdmMoodownJklS003 |
| Netsky.B | AdmSkynetJklS003 |
| Netsky.C-.E | [SkyNet.cz]SystemsMutex |
| Netsky.F, .K | LK[SkyNet.cz]SystemsMutex |
| Netsky.G | Netsky AV Guard |
| Netsky.H | MI[SkyNet.cz]SystemsMutex |
| Netsky.I | KO[SkyNet.cz]SystemsMutex |
| Netsky.J | SkYnEt_AVP |

The installation code follows next. This copies the worm in the %windir% (e.g. C:\Windows) directory with different names among the variants and links a registry key string to it, as listed below. The base for all registry keys RegBase is HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:

| | |
|---|---|
| Netsky.A, .B | RegBase\service<br>services.exe -serv |
| Netsky.C-.E | RegBase\ICQ Net<br>winlogon.exe -stealth |
| Netsky.F | RegBase\Zone Labs Client Ex<br>svchost.exe -antivirus service |

| | |
|---|---|
| Netsky.G | RegBase\Special Firewall Service<br>avguard.exe -av service |
| Netsky.H | RegBase\Antivirus<br>Maja.exe -antivirus service |
| Netsky.I | RegBase\Tiny AV<br>fooding.exe -antivirus service |
| Netsky.J | RegBase\My AV<br>avpguard.exe -av serv |
| Netsky.K | RegBase\ICQ Net<br>winlogon.exe –stealth |

## DISTURBING OWN KIND

After assuring self-execution at system startup, the worm disables various malware that it finds running on the machine – including variants of Bagle, Mydoom, Mimail and earlier variants of Netsky itself – by deleting their startup registry keys. While the first version of Netsky removes only several keys, variant .J erases more than two dozen:

Netsky.A and .B:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Taskmon
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Taskmon
HKCR\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\
InProcServer32
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Explorer
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Explorer
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
KasperskyAv
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
system.
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\
RunServices\system.
```

Netsky.C-.E and .K same as .A and .B plus:

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
KasperskyAv
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
msgsvr32
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
DELETEME
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
d3dupdate.exe
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
au.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
service
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
OLE
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Sentry
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\
PINF
HKLM\System\CurrentControlSet\Services\WksPatch
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Windows Services Host
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Windows Services Host
```

Netsky.F same as .C-.E and .K plus:

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
gouday.exe
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
rate.exe
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
sysmon.exe
```

Netsky.G-.J same as .F plus:

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
srate.exe
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
ssate.exe
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
sate.exe
```

## DISTURBING GODS

After self-installation the worm searches recursively on local drives from C:\ to Z:\ (but not on CD/DVD-ROM drives) for email addresses. Starting from the .C variant it initializes a list whose first entry is a hard-coded email address, which is different for each version. This helps the author to keep track of the propagation and distribution of the creation. However, there is a counter of harvested email addresses and when this counter reaches 10,240 (0x2800) the searching process is stopped. This limitation resides in the fact that the list is static and declared as

```
char *emailAddresses[0x2800];
```

It is at this point that P2P spreading is performed in those variants in which the function exists (as described earlier).

The following are the hard-coded email addresses used for monitoring, beginning with the .C variant:

| | |
|---|---|
| Netsky.C | jfkoofia@yahoo.com |
| Netsky.D, .E, .K | skoorpio@yahoo.com |
| Netsky.F | sko@yahoo.com |
| Netsky.G | mxfoi@mxc.com |
| Netsky.H | russia@yahoo.ru |
| Netsky.I | moobia@yahoo.com |
| Netsky.J | janette_james@yahoo.com |

Files that are scanned in the search for email addresses must have a specific extension, as follows:

For Netsky.A and .B:

| | | | | | |
|---|---|---|---|---|---|
| .xml | .wsh | .jsp | .msg | .oft | .sht |
| .dbx | .tbb | .adb | .dhtm | .cgi | .shtm |
| .uin | .rtf | .vbs | .doc | .wab | .asp |
| .php | .txt | .eml | .html | .htm | .pl |

For Netsky.C-.I and .K the list of extensions scanned is the same as for variants .A and .B, plus .cgi, .dhtm and .shtm. Netsky.J adds .jsp, .xml and .wsh to the same list as variants .C-.I and .K.

The worm has its own SMTP engine which it uses to send emails. One email is sent to each address that is found – at least as long as the infected system is not restarted. When the system is restarted the worm will search and find the same email addresses to which it has already been sent. This is because a flag is assigned to each email address, from a buffer of flags. The flag is set after the address has been used and is lost when the worm's process is terminated, as happens at system restart.

The worm contains certain hard-coded strings which are never used:

Netsky.A

```
#T#h#i#s# #i#s# #t#h#e#
#[#W#3#2#.#S#k#y#n#e#t#.#c#z#]# #A#n#T#i#V#i#R#u#S#
#-# #w#e# #w#a#n#t# #t#o# #k#i#l#l# #m#a#l#w#a#r#e#
#w#r#i#t#e#r#s#!#
```

Netsky.B

```
#n#o#t#n#e#t#s#k#y#-#s#k#y#n#e#t#!
```

Netsky.C

```
<-<- we are the skynet - you can't hide yourself! -
we kill malware writers (they have no chance!) -
[LaMeRz–>]MyDoom.F is a thief of our idea! -  -<
SkyNet AV vs. Malware >- ->->
```

Netsky.D, .E, .K

```
be aware! Skynet.cz - –>AntiHacker Crew<–
```

Netsky.F

```
Skynet AntiVirus -  Bagle - you are a looser!!!!
```

Netsky.G

```
Netsky AntiVirus -  Give up, bagle & mydoom, dude!
You are fucking your mother! I want to meet you in
the U,S.A, Road-App time enc:[fg.od.jgij], and the
you will know what pain is!
```

Netsky.H

```
Skynet AntiVirus -  MyDoom and Bagle are children
```

Netsky.I

```
Skynet AntiVirus -  MyDoom and Bagle are spammer
```

Due to the large number of variants and the consistent changes among them, there follows a brief description of each variant's peculiarities.

### Netsky.A

Being the first version of the worm, Netsky.A is far from perfect: everything is done manually – the program runs sequentially, with no threads; registry keys are removed à la 'copy-paste'. Netsky.A displays a message box entitled 'Error' with the message 'The file could not be opened' when the worm is run without any parameter to give the impression that an error has occurred and the program will terminate, but the worm continues its installation silently. The attachment name is chosen randomly from a list of 25 hard-coded names (a full list of these can be found at http://www.virusbtn.com/netsky.xml).

With a probability of 52 per cent a .zip file is created from the last 13 entries in the list. The worm will be stored in the zip without password and then sent in this form. The name of the zip file is the name from the table, with the double extension replaced by '.zip'. A bug exists in this function which means that the generated zip file sometimes contains the worm with the zip name.

The subject of the mail is 'Auction successful!' and the sender is spoofed randomly to one of the following:

EBay Auctions <responder@ebay.com>
Yahoo Auctions <auctions@yahoo.com>
Amazon automail <responder@amazon.com>
MSN Auctions <auctions@msn.com>
QXL Auctions <responder@qxl.com>
Ebay Auctions <responder@ebay.com>

The body text of the email is as follows, where *x* is a random digit:

```
#———— message was sent by automail agent ————#

Congratulations!

You were successful in the auction.

Auction ID       xxxxxxxxxxxxxxx
Product ID       xxxxxxxxxxxxxxx

A detailed description about the product and the bill
are attached to this mail.
Please contact the seller immediately.

Thank you!
```

### Netsky.B

Functionally, the .B variant of Netsky is the same as Netsky.A, the main difference between the two is that the emails sent by Netsky.B are more diversified.

The attachment name is made up of components chosen randomly: a base name plus a single or double extension. The base name is chosen from one of 40 words or word combinations (for the full list see http://www.virustbtn.com/netsky.xml#tab2). The first extension has a probability of 46.15 per cent of being present and is one of the following: .txt, .rtf, .doc, or .htm . The second is always present and may be .exe, .scr, .com or .pif. This results in 40 x 4 x 4 = 640 different filename combinations.

With a probability of 51.52 per cent a .zip file is created from the 640 mentioned name possibilities, in which the worm is stored without password and then sent in this form. The name of the zip file is the base name with the extension

.zip. In this case the probability of the first extension being integrated is 66.67 per cent.

The subject of the message is chosen randomly from nine variations, while the body is one of 47 variations. A full list can be found at http://www.virusbtn.com/netsky.xml#tab3.

**Netsky.C**

Improvements can be seen in the .C variant: the list of email addresses is initialized with a hard-coded address to monitor the worm's activity more effectively, the message box used previously has been removed and threads have been added. One thread collects email addresses, one is for the payload and when the harvest of emails has finished a third thread is created to send the emails.

Email harvesting has been improved by filtering the addresses to exclude those that contain the following strings:

| | | | | |
|---|---|---|---|---|
| icrosoft | antivi | ymantec | spam | avp |
| f-secur | itdefender | orman | cafee | aspersky |
| f-pro | orton | fbi | abuse | |

The domains of the emails are checked by resolving their addresses on different hard-coded DNS IPs, starting with the DNS of the local computer. By resolving the server names for the IPs found in the worm we observe that most of them are located in Germany. We quite confidently presume that the author is German, since timestamp values hint at this too.

The .C variant contains a large number of encoded strings – 262 message bodies, 100 base attachment names and 51 subjects. In 66.67 per cent of cases the subject of the mail is chosen at random from the list of 51 subjects, otherwise it is chosen (also at random) from the 262 hard-coded message bodies.

With a probability of 51.52 per cent the attachment is a .zip file. The base name for the attachment is in the form 'name' in 75.76 per cent of cases, while the remaining time it is in the form 'name1_name2', where name, name1 and name2 are taken from the list of 100 base attachment names. File extensions are appended mainly as in the .B variant. The sender of the mail is spoofed randomly as one of the harvested addresses.

The worm's payload consists of an emission of sounds from the system speaker. The noise is composed of tones in the range 0–3000Hz which last for 0 to 50ms, followed by 50ms of silence. This works out as an average of about 13 random sounds per second. The payload activates only on 26 February 2004, between 6:00am and 9:00am local time.

**Netsky.D**

Netsky.D is essentially the same as Netsky.C with some modified strings and minor code modifications. For example, 'messagelabs' and 'skynet' have been added to the list of strings to be avoided in harvested email addresses.

The major difference from Netsky.C resides in the fact that, besides the payload and harvest threads, Netsky.D creates eight threads for sending email. However, the number of stored strings used for subject, message body and attachment name has been greatly reduced.

The subject is chosen from 26 variations and is correlated with the attachment name. Another significant change is that the attachment can no longer be a .zip file; its name is picked from a list of 26 filenames. Finally, the message body in the .D variant is one of only six variations. Full lists of the subjects, attachment filenames and the message body variations for Netsky.D can be found at http://www.virusbtn.com/netsky.xml#tab4.

The payload for the .D variant activates on 2 March 2004 between 6:00am and 9:00am and is the same as for Netsky.C.

**Netsky.E**

From timestamp analysis, it appears that this version of the worm was released before Netsky.D. Detailed analysis confirms that it is more similar to Netsky.C than to Netsky.D.

Netsky.E still uses .zip files as attachments and has the long list of strings encoded, but at the same time it sends the emails on (only) four threads, has the additional two keywords for address filtering, namely 'messagelabs' and 'skynet' and the payload activates on 2 March 2004 between 6:00am and 9:00am.

**Netsky.F**

Netsky.F is very similar to Netsky.D, but with a few modifications: it sends emails on 16 threads and there are five new filter strings for email addresses: 'andasoftwa', 'freeav', 'sophos', 'antivir' and 'iruslis'. Its payload activates on 2 March 2004 between 6:00am and 9:00am.

**Netsky.G**

Netsky.G is very similar to Netsky.F; the differences are: the payload's date has changed to 10 March 2004 (but again, between the same hours), the attachment has a 50 per cent probability of being a .zip file and there are only eight threads for sending emails.

**Netsky.H**

Netsky.H is much closer to Netsky.F than to Netsky.G and, again, its timestamp hinted at this fact. The attachment cannot be a .zip file for this variant, the payload is set to 8 March 2004 between 11:00am and 12:00pm local time, but it sends emails on 32 threads.

**Netsky.I**

Although this variant was released before the .G variant but after the .H variant, Netsky.I has some interesting modifications. It does not create .zip attachments, activates on 5 March 2004 between 11:00am and 12:00pm and sends emails on eight threads only.

The sender is now spoofed as 'service@recipientdomain' and the attachment is constructed as 'http://www.recipientdomain/user/index.scr', where 'user' and 'recipientdomain' represent the recipient's name and domain taken from the email address. The worm chooses the subject randomly from a list of three strings with a corresponding message body:

| Subject | Message body |
|---|---|
| Mail account deactivated | Your mail account has been deactivated. To reactivate, follow the link. |
| Mail account closed | Your mail account has been closed. Click on the link for further details. |
| Mail account expired | Your mail account expired. Please follow the link to reactivate. |

**Netsky.J**

A few improvements have been added in this variant. For example, the following sequence of code has been added in 44 places:

```
pDummyString = "AntiHeuristicScan";
```

where 'pDummyString' is a global variable declared as:

```
char *pDummyString;
```

This has no effect on the worm's functionality, but helps avoid generic detection. New strings have been added for address filtering: 'noreply', 'automail' and 'responder', and emails are sent through eight threads.

The payload in this variant works a little differently. On 10 March 2004, between 10:00am and 11:00am the old noise is triggered. On 10 and 11 March 2004 an over-emailing function activates – the flag is no longer set for the addresses that have already been used. A user may continuously receive emails containing Netsky.J on those two days. Moreover, on 10 March 2004 it creates eight additional threads to send email, and on 11 March 2004 it creates 16 additional threads.

The email subject is chosen from one of three tables, each containing 26 strings. The first table contains strings of the form 'Re:... user...' and there is a probability of 30 per cent that the subject will be chosen from this table. The second contains strings of the form '... user..' and there is a probability of 33.33 per cent that the subject will be chosen from this. Finally, there is a 36.67 per cent chance that the subject will be chosen from the third table. 'User' is the recipient name extracted from the email address. The attachment name is picked from a list of 26 hard-coded strings, correspondent to the subject. The message body is chosen randomly from a list of 18.

On 13 March 2004 a message box is displayed every second. The box is entitled 'Information' and contains the text:

```
SkyNet has the full control of your system now
```

On 16 March, if the worm is still active, it listens on port 26 and when a connection is established the worm removes its startup registry key, displays the above-mentioned message box with the text:

```
Please remove the file avpguard.exe from your
Windows-Directory and do not open attachments
anymore. It can be a virus like bagle and mydoom or
similar malicious code. This is the Skynet-Antivirus!
```

and after that it quits.

The unused string within this variant is very long. It states that this is the last version of the worm and includes some personal notes to both AV and virus producers. It concludes:

```
the 11th of march is the skynet day
```

Certainly at the time of writing it seems that this was indeed the original Netsky author's last version, since Netsky.K was released before .F (and is almost identical to Netsky.D) and the next two variants, .L and .M, seem to have been compiled by a beginner on debug.

## REACHING THE SKY

The code analysed in the Netsky worm is a clear and structured one, being written by at least a medium-skilled programmer. But considering the evolution of the worm, its declared intentions and the influence on the virus industry we suspect that the author is someone who was very excited by his work and perhaps had too much spare time between his high school semesters.

The impact of this worm on the AV industry was large: besides the virus researchers' headaches, six variants were released in an interval of five days. And they came along with new versions of other malware (Welchia, Mydoom and Bagle) as responses to the Netsky threat which necessitated many emergency updates from AV companies per day.

However, this worm raised (again?) a big question and posed the challenge: how long will reactive protection stand against new, diverse and intensified malware released in the wild? While many AV producers added generic detection which proactively detected some of the new threats, we saw that a mild modification in the source code bypasses the generic routines/signatures.

# VIRUS ANALYSIS 2

## CHIBA WITTY BLUES

*Peter Ferrie, Frédéric Perriot, Péter Ször*
Symantec, USA

W32/Witty is a UDP-based worm employing a vulnerability in *ISS* security products, such as the *BlackICE* firewall, to spread. More specifically, Witty uses a stack buffer overflow in the code that parses ICQ v5 packets.

Witty is very similar to last year's W32/Slammer (see *VB*, March 2003, p.6) in a number of ways: it is short (only 647 bytes for the attack buffer, excluding the variable UDP payload padding), its sending rate is limited only by available bandwidth, and it selects random target IP addresses. Unlike Slammer, however, Witty features a very destructive payload: it overwrites random portions of the hard drives of machines it infects.

### THE PARSING EXPEDITION

The vulnerability used by Witty, discovered by *eEye*, was published on 18 March 2004 (PST). The worm appeared late on 19 March 2004 (PST). This is perhaps the shortest timespan we have experienced between the public announcement of a buffer overflow vulnerability and the spread of the corresponding worm. One day leaves little time for countermeasures to be deployed, especially since this vulnerability was announced on a Friday!

The bug in *ISS*'s software is in the Protocol Analysis Module (PAM) and is related to the parsing of ICQ v5 datagrams supposed to originate from ICQ servers. The PAM module is located in a DLL called iss_pam1.dll (for the *BlackICE* product). The module takes a UDP source port of 4000 as an indication that an incoming datagram is an ICQ server answer (it does so regardless of the destination port – this assumption is necessary, given the connectionless nature of the UDP protocol). Then the module parses the packet according to the ICQ protocol. If an ICQ v5 SRV_MULTI compound message is received, containing an SRV_USER_ONLINE packet followed by a specially crafted SRV_META_USER packet, it may result in a stack buffer being overwritten. Witty sends just such a message, overflowing an email address field, and hijacking a return address in a typical stack-smashing attack.

The SRV_USER_ONLINE and SRV_META_USER packets are obviously anomalous. In fact, they are not valid messages according to the ICQ v5 protocol. Instead, their fields are tuned to force a specific code path to be taken in the PAM module. The SRV_USER_ONLINE portion of the datagram contributes in setting up the parameters required for the PAM module to parse the malformed SRV_META_USER portion. The packet sizes are the exact minimum required. Most fields are zeroed out, but the spoofed user IP address is set to one in order to bypass a check for a non-zero value. To be able to produce such an optimised exploit in such a short time, it looks like the author enjoyed an uncanny knowledge of the inner workings of the vulnerable code.

### MONA LISA OVERFLOW

The overlong email address submitted by Witty (which gives the virus its name – it starts with an ASCII string that includes 'insert witty message here') is copied into a 512-byte stack buffer through an sprintf() call. After sprintf() returns, the function whose stack frame holds the overflowed buffer attempts to return to its caller. Since the return address is overwritten, the control flow is hijacked. Instead of returning, the function jumps to a 'jmp esp' instruction located at a constant offset in iss_pam1.dll. The 'jmp esp' instruction in turn transfers control to the top of the stack, which contains a backwards jump pointing to the beginning of the worm body.

The first step in the execution of the worm body is to initialize register edi to point to the beginning of the UDP packet payload. This register is later used when sending copies of the worm to new machines. Rather than rebuilding the attack buffer from scratch, Witty locates the original copy of its attack buffer on the heap. It does so by following a saved pointer on the stack frame of the procedure calling the vulnerable ICQ parsing routine. The obtained location is a pointer to the IP packet payload, so the worm then skips the UDP header by adding 8 to register 'edi'.

Witty then alters the stack pointer to avoid clobbering its own code by pushing data on the stack.

### KUANG EXPERT TYPE 11H

The worm functionality is fairly simple: it creates a UDP socket and binds it to port 4000, and starts sending copies of itself to random IP addresses, on random destination ports. Periodically, after every infection cycle (consisting of 20,000 attacks) it runs its destructive payload. Witty relies on the Import Address Table entries of iss_pam1.dll to call the kernel32 APIs it needs. It resolves the Winsock APIs dynamically.

The randomization is carried out by a Pseudo-Random Number Generator (PRNG) similar to the Linear Congruential Generator in Slammer (it uses the same multiplier and delta.) The PRNG is seeded with the value returned by GetTickCount() at the start of each infection cycle. As a result of a single PRNG being used to randomize

the target IP, destination port, and size of the UDP payload, these three parameters are correlated: 90 per cent of the IP address space will be attacked in a single way. To a given IP address in this space, the worm always sends a payload of a constant size to a constant destination port. This is independent of the attacking machine. The other 10 per cent of the IP address space may be attacked in two different ways (and no more than two). For instance, the IP address 209.134.161.35 may receive worm datagrams with the following characteristics:

209.134.161.35 attack type 1: 882 bytes to port 23280/udp
209.134.161.35 attack type 2: 1030 bytes to port 13615/udp

Occasionally, the PRNG of Witty will generate some IP addresses finishing in '.255'. Since the *eEye* advisory mentions explicitly that the vulnerability is exploitable by broadcasting malformed packets, it is legitimate to wonder whether the worm takes advantage of such a mass-propagation. Fortunately, the use of a socket to broadcast datagrams requires a specific option to be set, and the worm author did not take this step.

After every infection cycle, Witty attempts to overwrite a random 64k area of one of the first eight hard disks with the beginning of the memory image of iss_pam1.dll.

## PATTERN RECOGNITION

We believe that the variable destination port and UDP payload size used by Witty were designed by the author to evade IDS products. A consequence of the variable payload size is that additional padding is sent following the worm body. The padding just happens to be the content of heap memory after the IP payload of the worm packet, and as such it is variable. Thus the worm packets will not only have variable size, but also variable checksums in the general case. In addition, as we mentioned above, the target IP, destination port, and payload size are correlated, which may lead to confusion when looking at the worm traffic from a single IP (one may think that the UDP payload is a constant size).

All in all, Witty had some interesting characteristics that probably allowed it to fly under a number of IDS radars. It is likely that the author of the worm was familiar with IDS systems.

## BURNING CHROME

There have been several vulnerabilities discovered recently in security software, from bugs in OpenSSL (exploited by the Linux/Slapper worm) to ones in *Microsoft*'s ISA Server and there are surely more to come. Since security software layers are naturally at the front line of defence, vulnerabilities in them are hard to mitigate. Vulnerable security software may lull the user into a false sense of security: disabling network services and closing ports does not prevent the parsing of incoming data destined for these services. This is particularly striking in the case of Witty: most clients do not have any ICQ v5 client installed because the current ICQ protocol version is 8, and version 5 has been obsolete for years!

## COUNT ZERO

According to the *CAIDA* analysis of the spread of the Witty worm (http://www.caida.org/analysis/security/witty/), the early propagation of the worm does not match the expected rate of a natural infection. The pool of infected machines appearing in the first moments of the epidemic is too high to be explained by regular network scanning and exploitation. Instead, the *CAIDA* analysts propose that the population of the worm was seeded, by the worm's author injecting the worm code manually into a few pre-scanned vulnerable machines. We agree.

Rather than inferring this from epidemiological data, our evidence relies on the observation of a side-effect of the worm propagation method: when Witty sends itself to a target machine, it randomizes the size of the datagram it sends. The size of the datagram is between 768 bytes and 1279 bytes, therefore the datagram systematically includes a portion of unused data following the 647 meaningful bytes of the worm. Had the worm originated from a single instance, all of its replicants would share the same tail (between bytes 648 and 767). Such is not the case: firewall logs show that at least two different tails exist (there may be more such tails; the seeding population could be determined by counting them).

## CONCLUSION

Insert witty conclusion here. [*sic*]

### W32/Witty

| | |
|---|---|
| **Aliases:** | W32.Witty.Worm, W32/Witty.worm, WORM_WITTY.A, Worm.Win32.Witty. |
| **Size:** | Variable, between 768 and 1279 bytes for the UDP payload. |
| **Type:** | Internet worm. |
| **Exploits:** | Buffer overflow in ICQ parsing routine of the ISS PAM module CAN-2004-0362. |

# FEATURE 1

## PASSWORD-PROTECTED VIRUSES

*Dragos Onac*
BitDefender, Romania

Recently, new trends in virus writing have been observed, with viruses using clever spreading techniques to try to elude anti-virus applications. One such spreading method is via password-protected archives. This method may not be new (for example, it was used by W32/NoFear in 2003), but it has been popularized by several versions of the more recent virus W32/Bagle.

If a user is running an up-to-date anti-virus application, it should be pretty difficult to infect the machine, even if the anti-virus software is unable to decompress password-protected archives. But this is not the issue. The problem is that such archives should not have reached the user's computer. They must be intercepted on the mail or file server (whichever the user accesses).

Thus the problem remains: how to scan password-protected archives efficiently (quickly), before they reach and can be opened by the user.

### EXTRACTING PASSWORD-PROTECTED FILES FROM ARCHIVES

There are several solutions that can be implemented, each of them having advantages and disadvantages. The primary objective of these solutions is real-time scanning of encrypted compressed files.

### PASSWORD RECOVERY

Since the archive's password is usually sent inside the email body, there is a good chance that inspection of the message will reveal the password. This brings us to the problem of recovering the password directly from the mail body.

It is easiest to detect the password in plaintext messages, since (until now) common phrases are used to draw the user's attention to the password. For example:

```
the password is...
here is the password:
... is the password
```

This may seem very simple to a native English speaker, but if the writer's English is weak (or the writer wants it to appear that way), the message may be 'disrupted'; for example, "password" may be replaced with "pass-phrase" or "pwd", making it a little harder to detect the required information.

The next case is where the body of the email is sent as HTML or Unicode. The use of some simple parsers (which are usually included in an AV solution) allows us to reduce this to plain text. Of course, there is a small time penalty in doing so, but not a significant one.

In the third case, the email is sent as formatted text (for example RTF) but, again, the use of a suitable parser can reduce this to plain text without undue difficulty.

In some cases the password is sent with spaces (or dots, commas, etc.) between the letters. In this case, the user must input the password manually (copying and pasting the password will not work). Here, the probability of infection is reduced, because additional input is required from the user – as we have seen from other socially-engineered messages, the most effective way to spread is to require some user input, but not too much. Furthermore, string parsing techniques can be used to decipher the password.

The next method of sending passwords is using HTML messages containing javascript and/or vbscript code which will display the message. This method gives the user access to a password that can be copied and pasted and it will be significantly harder for an AV solution to detect the password, since script emulators will be needed. However, the majority of email clients no longer execute scripts embedded within email messages.

The next method involves sending the password in a format other than text. The simplest of these is to send the password as an image. This method is used by the newer versions of W32/Bagle and it seems to work pretty well (although it does not help to spreading the virus very much). In order to extract the password from the image, AV software needs to include image-recognition algorithms, which are time-consuming. Tests of several publicly-available optical character recognition (OCR) algorithms on Bagle's password images led to the conclusion that their detection rate is currently poor – but OCR can be used to enhance other methods. The speed of the algorithms is fair, but improvements are required if this method is to be used in a real-time scanning environment.

Sending the password as an image does make it harder for an AV solution to detect, but this also creates two major drawbacks for the malware. The first is the inability to copy and paste the string, which means that, again, a greater level of user input is required. The second drawback is the code that generates the image. If the virus writer tries to create the image from scratch (not using any of the available libraries on the host machine), the virus may become quite large. On the other hand, use of the host's libraries would make the virus dependent on a specific operating system, thus minimizing the risk of infection on platforms that do not support all the required functions.

The final method of distributing an archive password would be to use something like a riddle. This would be difficult for AV scanners to get around, because it requires the use of artificial intelligence algorithms. However, it is highly improbable that an average user would spend the effort attempting to solve a riddle – this is an extreme version of requiring user input. Furthermore, riddles may rely heavily on social engineering and individual culture. Even though the Internet has made communication simpler, it has not levelled culture such that millions of people could answer a simple question (riddle) with exactly the same words.

So, the retrieval of a password by AV software is hard, but not impossible, because the solutions preferred by the worms/viruses when sending themselves as an encrypted files are usually simple.

## BRUTE-FORCE

If extracting the password from the mail body fails, we have some other tools to try to break the password from the archive. One of these is the use of brute-force, generating combinations in a finite searching space and testing them against the encrypted archive. This is time-consuming, but imposing some strict limitations (such as don't try to brute-force large files, or limit the generator to the algorithms that already exist in viruses) leads to an efficient method of breaking archives. For example, the search time for Zip archives can be minimized by using the last byte of the 12-byte decrypted header of a protected file, which is known to be equal to the file's CRC. However, this only reduces the search space by 256 times – which, for a password containing eight characters, is insignificant in terms of the processing power required.

The disadvantage of this method is that it tests many passwords, so it's impossible to cover all the possibilities in a useful period of time. Enhancing it to minimize the searching space is a must, and this is highly dependent on the archive type in question.

## PLAIN TEXT ATTACK

Another way to extract files from a password-protected archive is to launch a plain-text attack against the archive. In this case, we try to 'guess' the key by testing every signature of viruses known to replicate via password-protected archives.

Using Zip archives as a test case (Zip archives possess one of the weakest encryption algorithms; see [1] and [2]), we have a searching space of 40 bits (2^40 possibilities) when using only three bytes for the plain text attack. When using 510 bytes (which is a pretty large signature), there are 2^30 possibilities. Usually, the password is found somewhere in the first quarter of the searching space, so this reduces the number of guesses to 2^38 bits (and 2^26 bits, respectively). The number of possibilities is very high, which leads to unusable scanning times for Zip files (other archives have stronger encryption algorithms, so the situation is even more complicated).

The plain text attack is, of course, dependent on the encryption algorithm, but one cannot afford to use it in a real-time environment; another drawback of this method is that it does not protect the user against viruses which are not known to replicate using password-protected archives (or against unknown viruses).

## DICTIONARY ATTACK

Using a dictionary and testing each word as a decryption key is limited in its efficiency, but this method can be used to enhance techniques such as brute force or password extracting. This method is not as time-consuming, so it can easily be used in conjunction with any of the previously described methods.

## CONCLUSIONS

Password-cracking for archives is not a simple issue, and it is no longer something that only hackers do. In time, it will be a requirement for every virus-scanning product. Even though it is not easy to implement a fast method of decrypting archives, it is achievable.

Of course, new methods of sending infected archives and their passwords may be invented, improved upon and updated, but for each of them, algorithms can be created that decrypt the viral code before the target user can even 'touch' the archive. Keeping in mind that virus writers are restricted by the need to create well socially-engineered messages, we reach the conclusion that most of the ways in which the password will be sent will be simple. And for simple problems, we can develop simple and fast solutions.

## BIBLIOGRAPHY

[1] *ZIP Attacks with Reduced Known-Plaintext*, Michael Stay, AccessData Corporation, published by +Tsehp March 2001.

[2] *A Known Plaintext Attack on the PKZIP Stream Cipher*, Eli Biham and Paul C. Kocher, Fast Software Encryption 2, Proceedings of the Leuven Workshop, LNCS 1008, December 1994.

[3] Linux and OCR homepage: http://www.linux-ocr.ekitap.gen.tr/.

# FEATURE 2

## HUNTING THE UNICORN

*Andrew J. Lee*
Independent anti-virus researcher, UK

One of the more prickly thorns in the side of the Anti-Virus world is the continual call for a unified or universal naming convention. This article looks at the hunt for a 'universal naming and identification convention that's really nice', hereinafter referred to as a UNICORN. The purpose is to examine the usual reasons why a UNICORN is sought, and why those are not necessarily the reasons one is needed. To provide a firmer basis than opinion for our postulation, we shall offer some survey data collated specifically to garner opinion about this subject. In the past there have been many attempts to define a UNICORN, with many different ideas about what a UNICORN really is. Rather than revisit old ground, we will try to look at what the need is, when it becomes a need and how the need can be addressed in a timely manner.

### WHY WE THINK WE NEED A UNICORN

Data for this research were gathered using a ten-question survey (available at http://www.linuxav.co.uk/survey.html), which was given to various interested groups. For each question a number of standard responses were offered, but there was also an opportunity to answer in the respondent's own words. It should not be assumed that this survey was particularly scientific, nor that it is necessarily unbiased and internally sound; it is, however, a good method of collecting interesting opinion, and gauging common reactions. In total 64 responses were received, of which 62 were 'valid'. The spread of respondents is shown in Figure 1.

The responses were grouped into three distinct categories, for further analysis.

- Group A contained corporate administrators (effectively end users with reasonably significant knowledge of the field) – this group constituted 37.1 per cent of the total respondents.
- Group B contained researchers including independent researchers and vendors and constituted 56.4 per cent of the total.
- Group C contained the others, either unidentified or not falling within the other groups and constituted 3.5 per cent of the total.

### NAMING THE TERMS

This seems an appropriate point at which to insert some definitions. By 'universal' we mean common to, or

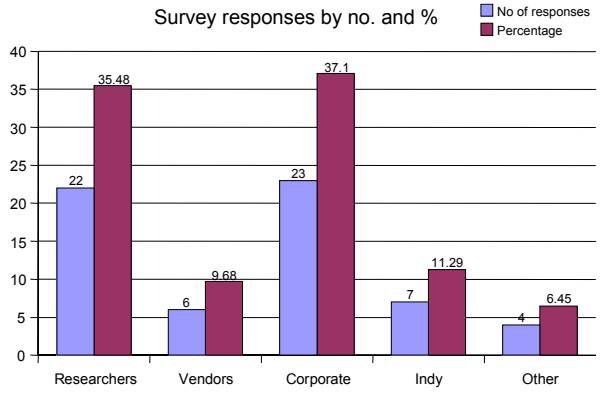



*Figure 1: Survey respondents.*

proceeding from all in a particular group. By 'convention' we mean an accepted rule or usage. By 'convergence' we mean to move or cause to move towards the same point. We use 'nice' in its less common meaning of 'precise', rather than 'pleasant' [1].

It is often assumed that, because everyone wants a UNICORN, they all want it for the same reasons. However, the results of the survey confirmed our suspicions that this is not the case. Indeed, it seems there may be as many reasons for wanting to find a UNICORN as there are individuals in the industry. However, there appear to be two main groups, each with its own requirements of a UNICORN. There may be several other groups – for example, AV-product testers, home users, journalists and security writers may each form a discrete group. (Certainly, there seems to be justification for treating the media as a separate group, given their propensity simply to make up or adapt their own virus names [Chernobyl, Matrix, Anna]). However, as representatives of these groups did not respond in significant numbers to this survey, it is not possible to draw any conclusions about them or their needs.

### CONVERGING ON A CONVENTION

The first question on the survey was the most obvious: "*Do you think a standardised virus/malware naming scheme would be useful?*" 98.4 per cent of responses were in the affirmative, with one person replying 'unsure'.

It seems safe to assume that, to the vast majority of this industry, naming is important, and a UNICORN is desirable. Often it is far easier to determine that something is desirable than to decide *why* something is desirable.

It is tempting to think that a UNICORN is about supplying informational needs, but this is no more true of computer malware than it is of biological ones. The general public does not need to know the exact taxonomy of, say, *Clostridium perfringens* or *Staphylococcus aureus*. They

realize that they have some sort of food poisoning – however, the treatment of one will not be the same as the treatment of the other. Indeed, the scientific names would often tell a sufferer less about the illness than the more common term 'food poisoning'. It could be argued that the only people who *really* need a naming convention for computer 'illnesses' are the people who write anti-virus products. The rest of us just need to know that what we're calling something is the same as everyone else is calling it – or at least have a method of cross-referencing different names. What this second group needs is not a naming convention – which refers to how a name is constructed, and of what elements it should consist – but a naming *convergence*; that is, the same name used by all.

*Nearly 84 per cent of respondents said they use three or more vendor websites for information gathering when assessing a new threat.*

One of the more interesting results from the survey was that nearly 84 per cent of respondents said they use three or more vendor websites for information gathering when assessing a new threat. This seems to indicate that there is no loyalty to a particular website as a 'one-stop-shop'. It seems likely that inconsistent naming is one of the factors contributing to this, aside from the obvious need for comprehensive threat information. This is backed up by the results garnered from the question: "*In their analyses, do you think more vendors should include the aliases (names) other vendors are using?*" There was a massive 93.2 per cent response in the affirmative, which backs up our earlier contention that, for many people, the major importance of a name is in knowing that we're all talking about the same thing.

This author made the classic mistake in a recent discussion of W32/Bagle.b; the majority of vendors were using the Bagle name, but one notable exception referred to it as W32/Tanx.a. In the midst of a heavy workload, and trying to determine the nature of the threat, I posted a statement to a closed list, to the effect that said vendor did not yet have detection. This was an easy mistake to make, perhaps, but it was one that caused some consternation, both for the members of the list, and for that vendor – all of which could have been avoided had there been a standard name.

## COUGHS AND SNEEZES

Most of the general public are content to say that they have 'food poisoning', or 'flu'; not many will know or care about the exact taxonomy of the actual bacterium or virus with which they are infected. They will also rarely understand that the first is usually caused by bacteria (there are some exceptions) treatable with antibiotics, and the second is caused by one of many thousands of viruses of the influenza family, none of which will be bothered by antibiotics. To a doctor this sort of information is the difference between effective treatments, and possibly harming the patient; to the patient, it's not often the primary consideration – "I'm ill, give me something to make it better."

There are clear parallels with the AV world. For instance, 'the public' often use the words 'virus', 'worm', 'Trojan', and 'bug' interchangeably when talking about malware, and this is the primary reason we are detecting adware, jokes, Trojans etc., none of which anti-virus programs were designed to be effective against.

To stretch a rather thin analogy slightly further, consider that Group B is comparable to the medical profession – they are the people who really need to know the taxonomy of the malware. This group requires a naming convention, not primarily because it's an acceptable way of transmitting information between its members, but because it requires specific information about a unique virus.

It should be noted that the most commonly used (or, at least, abused) convention, the CARO Virus Naming Convention [2], did not require convergence, only that certain elements must be present in a full virus name, and that specific rules be followed. For the most part, the researcher needs to know what something is (in far greater detail than the end user) so that, first, they can uniquely detect it and secondly, remove or disinfect it. This knowledge must then be incorporated into an automated product, which, especially for file disinfection, must be able to identify precisely what is virus and what is normal file, so that it can excise the infection correctly.

Very few non-vendor researchers write virus definitions, so why is it important for users to know the convention name? If we use a particular vendor, we rely on them to have accurate detection of all viruses, including new ones. The only time we really need to know a vendor's name is if a machine is infected and we need manual removal instructions.

## WHAT HOPE FOR THE REST OF US?

Unfortunately, a non-convergent naming convention is not particularly useful to Group A. Why? Simply because knowing the name – even the unified and unique name – of a virus tells you virtually nothing about that virus [3]. Question 5 of the survey asked: "*What is the most important part of a virus name?*" The options were: 'Type' (e.g. W32,

VBS), 'Family Name' (e.g. Klez, Sobig), 'Family and Variant' (e.g. Yaha.e), 'Attributes' (e.g. @mm) or 'Other'. 22 per cent of Group B thought 'Type' most important, as opposed to 39.1 per cent of Group A. This helps to demonstrate a difference in the needs of the two primary groups. Anti-virus researchers need to know which virus they are talking about, between themselves and other researchers, for the reasons described above. This is often simplified by the fact that they have access to a sample of the code, and usually can cross-reference information with their own research. If they need to talk to other researchers, they can refer to segments of the code, or if needed exchange the whole sample.

Corporate AV administrators want to know what the virus affects and what it is going to do. They don't necessarily have access to the code, and usually wouldn't have time (or in many cases the skills) to carry out a code-level analysis of a new threat. Certainly they would rarely have shared sample access with their vendor.

The first and most important question for any corporate AV administrator is "Is this a threat in my network?" Usually the only information they have in the initial stages of an outbreak is a name (or five).

Naming convergence between vendors is tremendously helpful here, in that it is easier to determine that there is only one threat that is being assessed (or that there are more than one). In these days of multiple rapid outbreaks, this is becoming increasingly important. This sort of dilemma can be partially addressed by vendors aliasing each other's names, and the use of VGrep (http://www.virusbtn.com/resources/vgrep).

However, to quote Nick FitzGerald [4]: "VGrep's very existence is evidence of an odd contradiction in this industry. The fact that it is needed is proof of how little the industry as a whole cares about naming consistency – if the industry really cared about naming, VGrep would not be needed." No doubt Nick would make a similar comment about aliasing – it's a cure for the symptom, not the disease.

## SMOKEY AND THE BANDIT

The speed at which malware can spread has increased exponentially in the last few years. In an 'always on' broadband world, the length of time that it will take a mass-mailing worm/virus to reach critical mass is equal to the time it will take for each of x number of emails to be delivered and read. Factor in how long it will take an AV company to be able to detect that malware. Even (dangerously) presuming that this is within minutes of the first non-researcher receiving it, or being infected by it, the time between initial receipt of malware and an update being available from that vendor's site (let alone installed on the customers' machines) will determine the time in which the malware can spread almost entirely unchecked [5]. In the constant chase between the cop 'Smokey' and 'The Bandit', the bandit is always one step ahead, and getting better at avoiding detection in those crucial early moments of an outbreak. In days that see three or four major fast-burners (say, Netsky, Bagle, Mydoom variants - see *VB*, April 2004, p.8) all coming out within hours of each other, a convergence in naming may be the only thing that can mitigate the panic.

## GRABBING THE BEAST BY THE HORN

If we eventually do find our UNICORN, who will referee the system?

It is clear that, market forces being what they are, a single vendor could not do this – nor would this be a good idea, as there would be clear conflict of interest with any vendor attempting to impose this. To date, CARO has made the most concerted effort both to provide and enforce (at least among members) a common naming convention (without requiring convergence between names). Unfortunately, this has been ignored deliberately by some in the industry [4]. It seems unlikely that this will ever be overcome – some vendors will still insist on doing their own thing, and some will do it just to spite the others. An independent body would be the ideal, but then one immediately meets the thorny issue of sample sharing – essential for anybody trying to determine that Virus A is the same as Virus B and should have the same name. There is also a time factor involved, and it is this, perhaps more than any other factor, that ultimately will scupper any attempt at having an immediate convergence of names. An accepted convention for naming may be more achievable, with eventual convergence where possible, and as soon as possible.

*The problem with proper naming is not so much that we couldn't do it, rather that it is very hard to do well.*

The problem with proper naming is not so much that we couldn't do it, rather that it is very hard to do well – so much so, that it is unlikely to be done at all and, because there is a split need between Groups A and B, the considerations of one are likely to be less important to the other.

The initial stages of an outbreak are necessarily when a virus is given a name (however temporary that name might be), and according to the survey results, this is also when a

UNICORN is most needed. We all want to know that we're talking about the same thing. If, as usually happens, more than one vendor makes simultaneous discovery of a new virus (often through customer submissions), they need to give it a name and, inevitably, this will differ between vendors.

There are two ways of dealing with this problem. Either no one names the virus at all, and each company submits its sample to a central body, which then cross checks and hands out a name; or everyone simply names it what they like, and sorts things out afterwards (or not). The problem with the first scenario is the time lag. The single most critical factor when dealing with a new virus is not deciding what it is called, but whether we can reliably detect it; this absolutely must be done as fast as possible. Arguably, in the time taken to do this, a sample could have been submitted, and a name decided upon, however, this is not necessarily the best way to approach the problem.

In the second scenario, we have the problem that we currently face; many vendors are simply not interested in renaming to suit any given convention. There are understandable reasons for this; it is a duplication of effort, for instance the detection patch must be updated – why update something that works already, when there are probably ten other viruses waiting for analysis? And, of course, there's plain old prickly pride – the 'we found it first so we're calling it xxx' syndrome.

Ultimately, I think there is a better chance of the second solution being the one that works. This will not fix the 'we need to know we're talking about the same thing' problem initially, but I think that problem is overplayed. There are now many ways of getting such coordination, for instance, the AVIEWS groups (http://www.aviews.net/) are ideally suited to cross-dissemination of information – and many vendors have taken to using 'alias' fields on their sites to show what other vendors are calling a virus.

## STONED MONKEYS

All this information is readily available at the moment, so we are left with the question, 'why do we need a naming convention?'. Perhaps because having on average four unique names for each virus [6] could be considered downright stupid? Perhaps, to quote one of the survey respondents "so we [AV researchers] don't collectively look like a bunch of gibbering monkeys, each thumping randomly at their own naming typewriter." Perhaps because it would just make life a whole lot easier when trying to find information.

What could work is for an independent organization to collate the information coming from each vendor into a central location – under a single name (for example http://secunia.com/virus_information/). Before my inbox catches fire, this name should be based on a standard convention, and should be the most commonly agreed name between the vendors, but it should also have a searchable version and revision history that will allow cross-referencing.

*We are probably further away from finding a UNICORN than we ever were.*

We are probably further away from finding a UNICORN than we ever were, partly because the less sure we are as to why we need it, or what we need from it, the less likely it is that we will ever define a usable schema. There are more viruses, more anti-virus companies, and outbreaks reach critical mass far faster than ever they did. The difficulties of trying to coordinate naming between so many different people – in a time that would be anywhere close to useful – are close to insurmountable. If we could get away from the idea that we need the name to tell us *about* the virus, rather than being a unique identifier of a specific infector, then we might get closer to building something that will actually do what we want it to. The name as such does not really matter much, after all – just as long as we all have the same information about the same thing.

## REFERENCES

[1] McLeod, W.T [Ed.] 1991, *The New Collins Dictionary and Thesaurus in One Volume*, Harper Collins.

[2] CARO, 1991, *A new virus naming convention*, see http://downloads.securityfocus.com/library/naming.txt.

[3] Gordon S., 2003, *Virus and Vulnerability Classification Schemes: Standards and Integration*, see http://securityresponse.symantec.com/avcenter/reference/virus.and.vulnerability.pdf.

[4] FitzGerald N., 2003, 'A virus by any other name – virus naming updated', *Virus Bulletin*, January 2003.

[5] Lee, A., & Harley, D., 2002, 'Back to the Future – Fresh Approaches to Malware Management', in Gattiker (Ed.), *EICAR Conference Best Paper Proceedings*, EICAR.

[6] Raiu C., 2002, *A Virus by Any Other Name: Virus Naming Practices*, see http://www.securityfocus.com/infocus/1587.

[*Next month, Vesselin Bontchev looks at the virus-naming problem from an AV vendor's point of view.*]

# PRODUCT REVIEW

## BITDEFENDER FOR SAMBA LINUX FILESERVERS 1.5.6-1

*Matt Ham*

*Softwin*, a Romanian company, has submitted its anti-virus products for *Virus Bulletin* reviews for a good few years and its product line has expanded notably during this time. In a market where the tendency has been towards expansion by acquisition, this company is remarkable in that it remains fully privately owned.

For most readers, *BitDefender* will be the product which springs to mind where *Softwin* is concerned. The company's range of products is markedly larger than this however. In terms of services, for example, *Softwin* offers *Contact Center* and *HelpDesk*. Other areas include SAP applications, XML and SGML development. While originally devoted entirely to anti-virus protection, the *BitDefender* product line has also been subject to expansion into related areas. Spam blocking is now supported, along with various methods of access control, including a personal firewall component.

*BitDefender* is available for a selection of *Windows*- and *Linux*-based platforms. The *Windows* versions cover the usual spread of desktop and server applications, together with *Microsoft*'s *ISA*, *SharePoint*, *Exchange 2000* and *Exchange 5.5*. *Lotus Domino* scanning is also supported, though only on *Microsoft* platforms.

On this occasion the *Red Hat Linux* version of the software was reviewed, specifically that for *Samba* servers. Versions also exist on the *Linux* platform for mail servers running SMTP, SendMail, Qmail or Postfix. According to the documentation, however, mail and *Samba* scanning cannot be operated from the same machine due to software conflicts.

In addition to these on-access server-based applications, there is a command-line version of the software which was also inspected. A bootable CD version of the *Linux* software is also available, bundled with a variety of system recovery and related tools. In common with other bootable *Linux*-based solutions this is also useful for data manipulation and scanning on *Windows*-based hardware.

### INSTALLATION AND UPDATE

Installation of the product is performed through a monolithic file – an .RPM within a .RUN wrapper. *Linux* products in general tend to be impressive in the length of their file names, this being no exception: 'BitDefender-samba-1.5.6-1.linux-gcc3x.i586.rpm.run'. This version is 3.1MB in size.

The .RUN wrapper is simply a shell script containing the .RPM, and is recognised as such by the *KDE Linux* interface, allowing it to be installed without having to open a console session manually. For those who prefer to interact with a command line, however, invoking using ./ works just as well. The shell script itself is easily visible if a user is interested in what is being performed by the wrapper.

Upon invoking the script, the licence agreement is the first significant information to appear, once the integrity of the archive has been checked. The licence is much the same as all others currently in operation for anti-virus software and, once its terms have been accepted, the installation proper commences.

There is no user interaction during the installation process, though a small number of messages are produced as the process is progressing. These are statements that the *BitDefender* engines and core have been installed, and that several objects have been registered. Following this, the installation is complete as far as the package is concerned. As far as the user is concerned, however, there is work still to be done.

As noted in the script, for each *Samba* share to be protected an appropriate line must be inserted in the /etc/samba/smb.conf file. *Samba* .conf files vary wildly in their complexity, with the *VB* test machines being very much at the simple, bare-bones end of configuration. This is not only because of the simple network in operation, but also because any added complexity gives an extra point at which sensitive anti-virus products may fail. As a side effect, this simplifies the addition of the required lines within this file, when installing such products as *BitDefender*. However, administrators with vast, epic and sprawling conf files may find the lack of automation here rather frustrating.

The line in question is, by default,

```
vfs object = /opt/BitDefender/lib/bdvfs2.so
```

and is stated explicitly at the end of the script processing. This is a good point as far as clarity is concerned, since the format of and requirement for this line is an area where many products seem to assume user clairvoyance. The line may be inserted as a [global] or on a per [share] based parameter, the former being rather simpler in large configuration files. There is not, however, a simple method for excluding one share from scanning while including all others – something which might be useful if, for example, central quarantining is required on a dedicated share.

The configuration of the scanning engines is also mentioned in the installation script output, this being controlled by means of an XML file. The user is referred to the manual for more details. Again, this information is provided in an obvious manner and will be useful to any first-time installer

of the product. Given *SoftWin*'s experience in designing XML for third parties, it comes as no surprise that this is the chosen configuration file format. Although rather unwieldy as a mode of control when direct editing is used, the configuration file may be manipulated through means of a *Windows* GUI. Hopefully a similar *Linux*-based configuration tool will be released at some point, whether graphical or command-line based. The XML file is a central location for all configuration options, not only of the *Samba* scanner but also reporting, updating and general daemon-related engine settings.

Installation location is not definable within a standard invocation, being to /opt/BitDefender/, with further directories within this being bin, etc, lib, share and var. There are, however, additional command line options for the install script, allowing a defined target directory, uninstallation, checksum authentication, information and a list of files within the installation archive. These are described if the script is invoked using the --? or --help command line modifiers.

Upgrading the product engine is a simple process, being performed simply by installing the new version over the old. By default, this will result in the deletion of all older files, though this can be avoided through use of command line switches.

Updates are slightly more complex and may be performed either manually or with a degree of automation through the *BitDefender Live!* application. Control of the automated process is through the XML configuration file. Updates are of two varieties, since, according to the documentation, update information is released fully on Mondays, with an incremental update on all other days.

Manual updating is a simple procedure, consisting only of unzipping the update files into the appropriate directory. At the time of writing, the situation with daily and cumulative updates was somewhat bizarre. Daily updates are available for the desktop product lines, yet for the server-based versions they are not. Moreover, the internal information for the update used noted it to have been produced on a Thursday, not a Monday, indicating that the documentation is not entirely accurate. The cumulative updates on offer currently stand at 2.3MB, which seems rather large when compared with the 3.1MB for the full product.

Updating automatically is performed with reference to the XML configuration file – with the default download location being upgrade.bitdefender.com. Upgrades and updates are also available from the *BitDefender* ftp site. Since the location of these files is configurable, it is also possible to set a local or network directory as the location from which information will be propagated – useful if one is wary of attaching servers directly to the Internet.

## WEB PRESENCE AND DOCUMENTATION

The primary web presence for *Softwin*'s security-related applications is at http://www.bitdefender.com/, with more general information about the company at http://www.softwin.ro/. The information and services available at the *BitDefender* website are nothing remarkable. Some sluggishness in page loading was noted, but no other problems were encountered.

As is becoming a more or less standard offering from anti-virus vendors, an online scanning option is available on the website. This was free from the delays encountered with the pages on the site – relying on the rather faster ftp site. It appears, however, that, in common with other such offerings, this service is available only on *Windows* machines.

Once installed, the sources of online documentation are twofold: files within the installation directory and command line help from the on-demand scanner. There are also manuals available for download from the *BitDefender* website. The web-based documentation is available either as a zipped 35-page PDF or in .tar.gz format.

Documentation as a whole appeared to have remarkably little overlap – the PDF documentation was devoted mostly to administration via the *Windows* configuration tool. The various readme files associated with the installation were, in general, of more use when performing activities directly on the *Linux* installation.

## SCANNING

As was noted in last month's *Linux* comparative review (see *VB*, April 2004, p.14), *BitDefender*'s command line scanner has certain quirks which made initial testing rather confusing. The first of these quirks is that the installation script does not set up paths or links to the on-demand scanning executable, which must be invoked explicitly with the correct path, or further manual configuring performed.

Performing a scan of the *VB* test collections was not without problems either. The selected extension list is configured such that only files with lower-case extensions will be scanned in the default mode. In the *VB* test sets, the file names are universally set as upper case in order to aid with logfile parsing. It was a pair of .pif files where the super-hidden extension was in lower case that hinted as to the reason why, out of the entire test set, only these two files were being scanned.

This is something of a concern, particularly in real-world situations, since in this case it is only file names which are causing non-detection. Turning on scanning of all files with a command line switch is recommended as a workaround.

The activation of scanning for all files is one of a number of command line switches available which allow for a certain degree of tweaking within the application.

The information concerning these options that was gained via the --help switch was not always as accurate as might be desired. The --nohed switch, for example, is advertised as 'unknown virus detection', which might be expected to allow heuristics. In fact, it suppresses the listing of heuristic detections in screen and log output, while retaining the overall numerical value of such files in the scan summary.

Likewise, there are typographical errors in the description of the switches that allow the copying and moving of suspected files to quarantine. With the use of a little logic, these are easy to ignore, since the switch --copys clearly activates copying of suspicious files to quarantine, rather than moving them as the description states. However, the quarantine function for suspicious files did not operate in a default installation.

As far as scanning results were concerned, the process was fast and detection rates generally high. Those areas where non-detection was a problem were related not to the newness of the samples involved, but rather their relatively odd nature. For example, the .HTM sample of W32/Nimda.A was missed, while versions with different extensions were detected flawlessly. This form of non-detection is usually remedied easily in future versions of a scanner.

On-access scanning will, by default, block all access to infected or suspicious files which are accessed through the shares where it is activated. Of course, this will have no effect on the transfer of infected files from one place to another on a server – only when files are accessed using the *Samba* protocol will there be scanning. This is an important distinction between the *Samba*-specific scanners and those which operate by scanning the entire *Linux* file system.

Where the *Samba* scanner was concerned, as opposed to the command line version, extensions proved to be of no importance when determining which files were to be scanned. On the other hand, the number of files to be scanned was of rather more importance than might have been assumed. When scanning particularly large numbers of infected files some instability of the *Samba* share was noted – an issue which was confirmed by subsequent investigation by the developers. This situation is scheduled to be ameliorated in the next scanner version.

The *Samba* scanner is integrated with a reporting system, which allows an administrator to gain an overview of what is occurring on his system. Likewise, it is possible to send anonymous reports to *Softwin*, detailing the viruses found on a particular system. In this circumstance the geographic point of origin is noted rather than any identifying information – the aim being to provide the developers with information on the prevalence and spread of malware. This reporting system is common to the entire *Softwin* range, and can be disabled if required.

## CONCLUSION

This review of *BitDefender* was mixed in many ways. In the comparative review, and previous *Linux* platform reviews, documentation was singled out as an area of weakness. As far as offline documentation is concerned *BitDefender* did not show any weakness at all, which can be considered a good thing. The command line help, on the other hand, was not so impressive.

Similarly, scanning itself was by and large good – both speed and accuracy were well within what can be considered the accepted norm for today's scanners. However, a number of quirks spoiled the overall effect. The extension issue on demand and the instability of the *Samba* scanner when faced with large collections of infected objects can certainly not be ignored.

When faced with mixed performance of this nature, the real issue is how likely such flaws are to be addressed, and how long they are likely to remain present. The second is easier to address – since the *Samba* product is still designated as being 'new' on the *BitDefender* website, these are unlikely to be long-standing issues. Given the steady improvements seen in the *Softwin Windows* product lines in the past, it is likely that these issues will be addressed sooner rather than later.

**Technical Details**

**Product:** *BitDefender for Samba Linux Fileservers 1.5.6-1.*

**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 *Red Hat Linux 9*, kernel build 2.4.20-8 and *Samba* version 2.2.7a. An additional machine running *Windows NT 4 SP 6* was used to perform read operations on the *Samba* shared files during on-access testing.

**Developer:** *Softwin SRL*, 5th Fabrica de Glucoza St., Bucharest, Romania; email sales@bitdefender.com; website http://www.bitdefender.com/.

## ERRATUM: RED HAT LINUX COMPARATIVE REVIEW, APRIL 2004

Due to the use of an earlier product version (1.5.5-2) in *VB*'s recent *Linux* comparative review (see *VB*, April 2004, p.14), the In the Wild detection figures published for *BitDefender for Samba Linux Fileservers* were slightly lower than should have been the case. The figure given in the review was 99.12%. This should have read 99.93%. *Virus Bulletin* apologises for the confusion.

# END NOTES & NEWS

**The VII Ibero American Seminar of Security in Computer Science Technology will be held 10–15 May in Havana, Cuba**. Topics will include: security on servers and network servers, database security, computer forensics, cryptography, intrusion detection, authentication and control mechanisms, policies and standards of security, ethics and legal aspects of computer security. For more information see http://www.informaticahabana.com/.

**The Black Hat Briefings and Training Europe takes place 17–20 May 2004 in Amsterdam, The Netherlands**. For more information see http://www.blackhat.com/.

**RSA Japan takes place 31 May to 1 June 2004 at the Akasaka Prince Hotel, Tokyo**. For details see http://www.rsaconference.com/.

**The Sixth Annual International Techno-Security Conference will be held 6–9 June 2004 in Myrtle Beach, CA, USA**. Topics will include computer forensics, Homeland security, intrusion detection, 'street smarts for cybercops', technical counter-terrorism, privacy issues, and security policies. For full details see http://www.technosecurity.com/.

**The 10th Annual Gartner IT Security Summit takes place 7–9 June 2004 in Washington, D.C., USA**. Topics include critical infrastructure protection, securing the workplace, security software and security strategies. See http://www3.gartner.com/.

**NetSec will take place 14–16 June 2004 in San Francisco, CA, USA**. The conference programme will include management issues of awareness, privacy and policy as well as more technical issues such as wireless security, VPNs and Internet security. See http://www.gocsi.com/.

**Internet Security & Payments takes place 15–17 June 2004 in London** as part of the Internet World UK event. For details see http://www.internetworld.com/.

**MIS Training will host a CISO Executive Summit in Geneva on 16 and 17 June 2004**. This event for IT security leaders will cover the unique issues faced by CISOs. For more information contact Yvonne Hynes on +44 20 77798975 or email yhynes@misti.com.

**The ISACA International Conference will be held 27–30 June 2004 in Boston, Massachusetts, USA**. Designed for professionals responsible for IT assurance, security, control and governance, the conference will provide in-depth coverage of solutions to technical and managerial issues. See http://www.isaca.org/.

**The Black Hat Training and Briefings USA take place 24–29 July 2004 in Las Vegas, NV, USA**. The call for papers remains open until 1 June, 2004. For full details see http://www.blackhat.com/.

**The 13th USENIX Security Symposium will be held August 9-13, 2004, in San Diego, CA, USA**. For details see http://www.usenix.org/.

**The 19th IFIP International Information Security Conference (SEC 2004) takes place 23–26 August 2004, in Toulouse, France**. Topics include intrusion detection, security architectures, security verification, multilateral security and computer forensics. For more information see http://www.laas.fr/sec2004/ .

**The ISACA Network Security Conference will be held 13–15 September 2004 in Las Vegas, NV, USA and 15–17 November 2004 in Budapest, Hungary**. Workshops and sessions will present the program and technical sides of information security, including risk management and policy components. Presentations will discuss the technologies, and the best practices in designing, deploying, operating and auditing them. See http://www.isaca.org/.

**The 14th Virus Bulletin International Conference and Exhibition, VB2004, takes place 29 September to 1 October 2004 at the Fairmont Chicago, IL, USA**. For more information about the conference, including online registration, the full conference programme and details of exhibition opportunities, see http://www.virusbtn.com/.

**The 31st Annual Computer Security Conference and Expo will take place 8–10 November 2004 in Washington, D.C., USA**. 14 tracks will cover topics including wireless, management, forensics, attacks and countermeasures, compliance and privacy and advanced technology. For details see http://www.gocsi.com/.




## SUBSCRIPTION RATES

**Subscription price for 1 year (12 issues) including first-class/airmail delivery:** £195 (US$310)

**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 www.virusbtn.com

# vb Spam supplement

CONTENTS



# NEWS & EVENTS

## FIGURES AND FLUCTUATIONS

While the term 'spam' celebrated its tenth birthday last month, *MessageLabs* reported a decline in the volume of spam seen in March 2004. However, a spokesman for the company thought this was unlikely to be a continuing trend, and pointed out that, historically, similar slumps have been followed by even greater surges in spam volumes. Meanwhile, the managed email security company has reported an almost 800-fold increase in the number of phishing emails in circulation over the last six months. Spam messages that tip shares have also been on the increase over the past few months according to UK-based anti-spam firm *Clearswift*. The company noted an increase in such emails between December 2003 and March 2004.

A recent report by analyst firm *IDC* indicates that, in 2003, spam accounted for 32 per cent of email sent per day in North America. As is the wont of analyst firms, *IDC* performed an impressive amount of number-crunching, to arrive at the conclusion that a company with 5,000 employees (but no anti-spam solution) would be set back US$4.1 million per year, as a result of each employee having to waste 10 minutes per day dealing with spam.

## SPAM ON THE CATWALK

Scott Richter, spammer extraordinaire, is set to launch his own clothing line, according to *Newsday* (see http://www.newsday.com/). The man who brought to our inboxes such 'unbeatable' offers as the set of Iraq's Most Wanted playing cards for only $5.99 (and who reportedly sold 40,000 decks in the first week), is planning the launch of the 'SpamKing' clothing line – to include hats, shirts and underwear. According to *Newsday,* the clothes will sport slogans such as 'Just opt out' and 'Click it'. Richter, who is currently being sued by the New York attorney general for spam-related crimes, plans to sell the clothes online and advertise – you've guessed it – by email.

## BUSINESS NEWS

UK-based web and email filtering company *SurfControl* revealed plans last month to acquire California-based email security firm *MessageSoft Inc*. The company will pay $9.69 million for *MessageSoft* initially, with an additional $5 million to be paid by the end of the year, dependent on sales. In acquiring *MessageSoft*, *SurfControl* will gain access both to the American company's technology and to its Chinese sales and distribution operation.

Small and medium-sized enterprises (SMEs) appear to be flavour of the month amongst anti-spam and security firms at the moment. While *Sophos* launched its anti-virus and anti-spam solution for SMEs – *Sophos Small Business Suite* – in the middle of April, *CipherTrust* and *Mirapoint* chose to launch their SME products at the London Infosecurity exhibition at the end of the month. Even *IBM* has joined the SME party, announcing its managed services for small and medium businesses which include scanning incoming and outgoing emails for viruses and spam.

## LAW AND ORDER

Spamming is about to become illegal in The Netherlands, after the upper house of the Dutch parliament approved changes to telecoms legislation late last month. However, the legislation – which is likely to become law in May or June – only covers spam sent to individual users; spam sent to businesses is unaffected. Under the new law unsolicited commercial email may not lawfully be sent to consumers unless they have opted in to receive messages. The Dutch Minister of Economic Affairs, Laurens Jan Brinkhorst, has answered critics concerned about the lack of protection afforded to businesses, by promising to undertake 'initiatives' to ensure that anti-spam protection is extended to the business community.

In other legislative news, the Federal Spam Act 2003 took effect in Australia on 11 April. Companies found to be in breach of the legislation will be fined up to AU$220,000 per day or AU$1.1 million for repeat offences. The Australian Computer Society has released a five-step guide for businesses to make sure they are compliant with the new legislation.

In Maryland, USA, an anti-spam bill has been approved that calls for stiff financial penalties and up to 10 years imprisonment for spammers. If the bill is signed into law, spammers sending more than 10 unsolicited commercial emails per day to users in Maryland could face a $10,000 fine or up to five years imprisonment, while repeat offenders could be looking at a 10-year sentence and a $25,000 fine. Currently, the anti-spam law in Virginia is considered to be the toughest (in the US) on spammers, but Virginian spammers can send up to 10,000 messages before facing charges. Meanwhile, in Virginia, the counsel representing a man charged with spam offences has requested that the attorney general's office be disqualified from prosecuting the case because the attorney general received campaign funds from *AOL* – the plaintiff. Furthermore, the defendant's legal advisor has called for the case to be dismissed since, he argues, the Virginian anti-spam law violates the federal Commerce Clause and the First Amendment. The trial date is set for September.

## EVENTS

Email security and management company *Postini* is to provide an on-going series of free webinars on how to eliminate spam and other threats in corporate email systems. See http://www.postini.com/.

The Messaging Anti-Abuse Working Group (MAAWG) Summit will be held 18–19 May 2004 in Washington D.C., USA. See http://maawg.kavi.com/.

INBOX – The Email Event will be held 2–6 June 2004 in San Jose, CA, USA. The event will be dedicated to all aspects of email, with 50 conference sessions covering subjects including security and anti-spam, as well as an accompanying exhibition. See http://www.inboxevent.com/.

The Institute for Spam and Internet Public Policy's (ISIPP) International Spam Law & Policies Conference takes place 29 July 2004 in San Francisco, CA, USA. Delegates planning to attend the CEAS event (see below) save $50 on the cost of registration for the ISIPP's conference. Details can be found at http://www.isipp.com/events.php.

The first Conference on Email and Anti-Spam (CEAS) will be held 30 July to 1 August 2004 in Mountain View, CA, USA. Further details can be found at http://www.ceas.cc/.

# FEATURE

## CLOSING LOOPHOLES: MTA MARK AND SPF

*Pete Sergeant*

By design, SMTP allows any server to transmit messages which it may claim originate from any user. Nothing, right now (except perhaps the threat of litigation) prevents you from sending friends an email that looks like it comes from 'bush@whitehouse.gov', or 'billg@microsoft.com'. In addition, MTAs are designed not to be picky about which IP addresses they accept email from – to most MTAs, the IP address of a machine running off a cable modem looks identical to that of a major mail exchanger for a large ISP.

Both of these features of SMTP are also weaknesses that allow spam to flourish. While they bring equality to all users of the Internet, and make life easier for those who want to send work-related email from a computer connected to the Internet via a cellphone, this equality is also afforded to spammers.

This article looks at two proposed filter-aids, one that tackles the problem of recognising whether a machine should be sending email directly, and one that helps authenticate that a given machine should be sending email for a given host. Both systems should make it harder for compromised machines to send viruses and spam – the machine would be forced to use the ISP's SMTP server to send email, which affords the ISP a lot more control over the content it sends out. A rate limiter at the ISP's end could dramatically reduce the utility of a machine for a spammer, while legitimate users remain almost completely untouched.

### MTA MARK

MTA Mark is a very lightweight extensible system for determining whether an MTA should be running on a given IP address. It sits on top of the DNS, and allows the owner of an IP address (the organization or ISP to whom the IP is assigned) to set a simple flag to reflect this.

More and more spam comes from 'zombie' machines these days – desktops connected to the Internet with a broadband connection, left on all day, and exploited by spammers to send spam. MTA Mark allows the service provider to specify that these machines should not be sending email directly, and thus helps mail recipients to filter out email that has been sent from 'unauthorized' machines.

DNS is a tree-based hierachy which allows domain administrators to delegate information about subdomains to other people. When trying to locate information about the host 'www.virusbtn.com', you first query the root name

servers for information about the domain 'com', which tell you where to find information about 'virusbtn', and so on.

There exists a domain specifically for looking up hostnames from IP addresses. Exactly how this works is beyond the scope of this article, but you can look up the hostname associated with an IP address by reversing the IP address to a root-to-last format (like a hostname), and then appending the domain 'in-addr.arpa'. Thus, in order to look up the hostname for 193.149.44.136, we look for records associated with the hostname 136.44.149.193.in-addr.arpa.

MTA Mark builds on top of this by using subdomains of this domain to store attributes of the domain in a tree structure with the use of TXT records. So:

| | |
|---|---|
| `[...]` | |
| `136` | last quad of IP |
| `_srv` | attributes relating to services |
| `_smtp` | where the service is SMTP |
| `_perm` | and we want to know whether it is permitted to run on this host |

Time for an example. The last email I received from Markus Stumpf, one of the creators of MTA Mark, was delivered to my mail server from the IP address 195.30.1.100.

First, let's look up the hostname of this machine:

```
~$ host -t ANY 100.1.30.195.in-addr.arpa.
100.1.30.195.in-addr.arpa   PTR  moebius2.Space.Net
100.1.30.195.in-addr.arpa   RP   abuse.Space.Net
```

So the machine from which I received the email was called moebius2.Space.Net, and the person in charge of it can be reached at abuse@Space.Net. So far so good, but was the machine allowed to send me email, as far as the ISP/organization in charge of the IP is concerned?

```
~$ host -t ANY _perm._smtp._srv.100.1.30.195.in-addr.arpa
_perm._smtp._srv.100.1.30.195.in-addr.arpa   TXT   "1"
```

And we get back a true answer, which means 'yes'.

MTA Mark's simplicity and lack of 'bloat' make it attractive for ISPs to implement – very little work is needed on the part of an ISP to 'blacklist' all their home-users very quickly. The fact that it has been designed with extensibility in mind also makes it very attractive – the domain-tree attribute system in the in-addr.arpa domain space allows all sorts of interesting possibilities.

## SPF (SENDER POLICY FRAMEWORK)

SPF goes one step further than MTA Mark and attempts to verify that the server transmitting an email is authorized to transmit mail for the domain of the apparent originating user. This makes it useful both for identifying spoofed email from viruses and spam-Trojans, and for identifying joe-jobs and forged email. As can be imagined, its format is somewhat more involved than that of MTA Mark.

From a purely DNS point of view, SPF is simpler: the SPF data is held in a TXT record associated with the sending domain. When you want to validate that an IP address is allowed to send email for a given domain, you grab the TXT records associated with that domain, and look for one that starts 'v=spf'. The exact SPF specification is beyond the scope of this article, but let's look at some simple examples.

I send my personal email from a domain called 'clueball.com'. All the mail I send is via one of two co-located machines, which, incidentally, are also the listed mail receivers, or mail exchangers (MXs) for the domain.

Situation: Any server listed as an MX server is allowed to send mail too

Translation: `v=spf1 mx`

*Hotmail* is, debatably, the largest provider of free web-based email in the world, and also one of the most forged domains in spam. Assuming (quite reasonably) that all *Hotmail* SMTP servers have hostnames that end in 'hotmail.com':

Situation: Any server whose IP reverses to a hostname that ends the with the domain in question

Translation: `v=spf1 ptr`

As you can see, SPF allows for exceptional brevity. And you can save yourself the hassle of learning the entire SPF specification by using an online wizard to generate your SPF records: http://spf.pobox.com/wizard.html.

SPF is rapidly being adopted, which, considering it is an open and patent-free solution to a problem that others (*Microsoft* and *Yahoo*) are trying to solve in a less-than-free way, is a good thing. Current users listed on the SPF website include *AOL*, *Google* and Oxford University. *SpamAssassin*, the venerable open-source anti-spam product will be filtering using SPF in the next major release.

Complete adoption of SPF will not be completely painless, however. People who use forwarded email addresses for their primary email address and people who send email directly from their laptop on the go, could face problems. Some common objections and solutions are listed here: http://spf.pobox.com/objections.html.

## CONCLUSION

Retrospectively tightening up a protocol that has been in use for an Internet lifetime, and is central to what we consider 'the Internet' is a non-trivial task. Eventually, maybe something will replace SMTP, but until then, closing loopholes in the mail flow may be one of the most effective anti-spam mechanisms we have.

# SUMMARY

## ASRG SUMMARY: APRIL 2004

*Pete Sergeant*

Summarizing ASRG is normally a matter of sifting through several hundred emails. This month, however, there was considerably reduced traffic – perhaps a sign that we're finally getting on top of the spam problem? When you've finished laughing …

The subject of some form of micropayment for sending email reared its ugly head once again, and was met with a long and detailed rebuttal from John Levine. He started with the point that no postage payment scheme has ever existed to deter use, rather than to recoup the cost of delivery. He then pointed out that the process of validating postage – checking that a given 'stamp' has not been used before, checking that it is not completely bogus – would require an exceptionally expensive infrastructure.

John pointed out that settlement is an administrative nightmare: with a central settlement company, you have single points of failure, if it is done at the ISP level, you have to interact with all the other ISPs to settle up – this is impractical. He then tackled the issue of exploited machines being used to send spam:

"The response I usually hear from e-postage enthusiasts is that Aunt Betsy won't let the zombies on her PC once she's had to pay a few hundred bucks in spam e-postage. Based on my observation of the real world, that's not gonna [*sic*] happen. Every month you see the predictable story about some loser whose PC got misconfigured or got a Moldavian dialer installed or something, and was shocked to get a thousand dollar phone bill. Do they actually pay the thousand bucks? Never. They negotiate it down, stiff the phone company, or something. ISPs would be stuck in a no-win situation where their customers will hate them if they try to collect, and their email peers will hate them if they don't."

He went on to debunk links to industries like the phone industry, by pointing out that the market is different altogether (email delivery is almost 'perfect competition', where the telecoms industry is an oligopoly, with high barriers to entry, and heavy regulation), and finished off by stating that the environment that an e-postage system faces is not one with a little fraud, but one with vastly more attempted bogus transactions from spammers than real ones, yet the cost of verification falls on the users, not the spammers.

Anne Mitchell updated the list members on the current status of the IADB, a database listing the adherence of certain registered hosts to a set of 'responsible mailer' criteria. The database now contains data on whether the listee publishes SPF records, whether they publish MS Caller ID records, whether they participate in the Bonded Sender Program, and whether they use *Habeas*.

The ISIPP (Institute for Spam and Internet Public Policy) has also launched the ISIPP Domains Database. This is a companion database to the IADB, which serves as a central registry to perform a very similar job to SPF – to specify which IPs can send email that claims to come from a given domain. Anne Mitchell was keen to point out that this is not intended to replace SPF, MS Caller-ID or friends, but that the ISIPP is merely responding to the demands of its customers – perhaps it was this disclaimer that stemmed the tide of anti-centralised accreditation rants one might have expected at this point.

Yakov Shafranovich posted a link to SURBL, a real-time blacklist based on the hostnames of 'spamvertised' websites: http://www.surbl.org/.

Alan DeKok highlighted the fallibility of RBLs owing to the fact that they are run by humans, and talked about how politics can often take over – RBL administrators blacklisting each other and the like. He gave a specific example from the ASRG itself, and likened it to censorship, rather than anti-spam filtering.

'der Mouse' made the point that there needs to be some level of accountability for users with exploited machines, probably in the form of a non-trivial cost. In der Mouse's own words: "Same as if Aunt Mary doesn't lock her car and someone swipes it for a joy-ride; Aunt Mary pays (in various ways, not all monetary, and not necessarily equal to any damage done, but, she pays)."

Yakov Shafranovich indicated that the US Government agencies have been busy this month, with the FCC having published details of its proposed rule-making for wireless spam, and the FTC adopting a rule that requires spam containing sexually-explicit material to be labelled as such. As expected, this was greeted by a fair amount of derision from participants on the list. John Levine valiantly defended the people working at the FTC, saying they are exceptionally clued-up and competent, however, they have to do as directed by the elected officials who are their bosses.

Most people seemed to miss the point that this new regulation would push even more spam into the realms of illegality, so that when the US Government finally gives the FTC the resources to prosecute spammers, it'll be easier to find charges to stick on them – a large cash injection into the FTC, to allow them to prosecute known spammers, remains this summarizer's favourite Final Ultimate Solution to Spam.