ISSN 0956-9979 MARCH 1995

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Richard Ford**

Technical Editor: **Fridrik Skulason**

Consulting Editor: **Edward Wilding**, Network Security Management, UK

CONTENTS

IN THIS ISSUE:

• **New threats.** Last month's editorial discussed the risks inherent in *Windows NT*. *Virus Bulletin* commissioned a series of tests on what happens if an *NT* machine becomes infected with a DOS boot sector virus: the results, and a discussion of how one can minimise the threat, are given on p.14.

• **Microsoft's free gift.** With every demonstration copy of *Windows 95* given out at a developers meeting in London last month came a free copy of the Form virus. See page 3.

• **Virus protection with a difference.** *Thompson Network Software's Network Security Organiser* takes a slightly different approach to virus prevention than the usual *NetWare* Loadable Module. A discussion of its capabilities can be found on p.18.

EDITORIAL

Mud Thrown…

One of the biggest misconceptions under which everyone in the anti-virus industry labours is that the virus authors and the virus hunters are in fact the same people. Although this myth is slowly being dispelled, it is still extremely important that the industry is seen to be ethical. While dialogue with proponents of virus distribution is important (indeed it may be the only long-term way to reduce the flood of new viruses), the relationship between virus authors and anti-virus software developers is a complex one; the fact that without viruses there would be no need for anti-virus software is self-evident.

However, it is all too easy to draw spurious conclusions from this statement. One often-heard argument is that virus writing is encouraged (or even carried out) by those in the industry, to keep themselves in a job. This is as obviously ludicrous as arguing that the police support or encourage crime in order to give them something to do! Nonetheless, this suspicion, this hint of a conflict of interests, has dogged anti-virus software developers for years. This is somewhat counterproductive, and gives the industry a rather shady and unsavoury image.

“ This ‘response vacuum’ allows wild claims to tarnish the reputation of all anti-virus software manufacturers ”

Another argument which is at first glance seductive (although it rapidly becomes less so when one examines most examples of virus code in detail) is that it only makes sense for those who write viruses to develop solutions to them. Certainly, if one were having problems with a particularly complex piece of virus code, the best person to approach for advice on how to deal with it would be the virus author himself, although this oversteps the line between what is acceptable behaviour and what is not: those who create the problem ought not be the same people as those who produce (and profit from) the solution.

Few would dispute that the customer deserves the best quality anti-virus software possible, but just how far should a company go to ensure that its product is the best? Would it be acceptable to strike a deal with certain virus writing groups so that they were paid a sum of money if they sent their viruses directly to developers? Obviously not. Is it acceptable to pay a virus author to develop a solution to one of his own viruses? Again, obviously not. Purchase new viruses direct from the author? Certainly not. However, there is a grey scale of lesser indiscretions which will arguably lead to better products in the short term. The question of what is acceptable and what is not is one which at present lies completely with each individual company.

A perfect example of this type of difficult choice was brought up when Mark Ludwig launched his CD-ROM virus collection (see *Virus Bulletin* June 1994, p.2, and July 1994, pp.6-7). Vendors were torn between buying the collection and possibly encouraging further updates, and not buying the collection and providing what could be inadequate protection for their customers. Unfortunately, there was no coherent stance taken, and any impact which the industry response might have had was ruined by everyone running off in different directions.

One of the things the industry lacks is a central governing body which can help regulate the actions of its members. Currently, there is little or nothing to stop any company doing whatever it pleases within the law, even writing its own viruses. The need for a central body is clear, yet due to the nature of the industry unlikely to be satisfied, at least in the foreseeable future.

The absence of an industry regulator is most keenly felt when individual developers come under fire for breaching accepted ethical standards. Such actions range from hyping the problem for short-term gain to claims of employing known virus writers. When such a situation arises, there is no coherent response, nor any formal statement on what we as a group find acceptable. This ‘response vacuum’ allows wild claims to tarnish the reputation of *all* anti-virus software manufacturers, whether or not such claims have any substance. Equally, it allows those who want to make money and do not care how they do it to act as they wish. Innocence or guilt is of secondary importance; all that matters is how the public perceives the situation. Unless strenuously wiped off, such mud sticks… however, it sticks to us all, not just to its intended target. Acceptable? One hopes not.

NEWS

Tall Tales

Mark Ludwig is determined to do all he can to keep *American Eagle Publications* in the public eye. The latest offering from the company is *The Virus Creation Labs: A Journey Into the Underground*, a book by *Crypt Newsletter* editor, 'George Smith'. Predictably, the book makes several claims about various companies and individuals associated with anti-virus software development or distribution.

In one extract posted on the *Internet*, it is claimed that young American virus writer James Gentile was employed by *Norman Data Defense*, although the company knew of his activities. Gentile, reputed to be the author of Satanbug and Natas (Little Loc), was to deal specifically to deal with these viruses.

Gunnel B Wullstein, director at the company's Norwegian headquarters, was quick to deny that this was ever the case, although she did confirm that there had been some contact between the company and Gentile over a period of only two weeks in the summer of 1994. She stated categorically that Gentile was at no time formally employed by them.

'It is important,' Wullstein said, 'to have a clean past in a field such as this. For us, his past was simply too close. Contact with virus authors or hackers is nothing exceptional in our business … The important issue here is that no company data or program information is jeopardized, and that no access to critical data is possible.'

She went on to say: 'We confirm that there has been contact with James Gentile … This contact did of course in no way set our products or development in danger.' ∎

Microsoft Distributes Infected Disks

After a developers' meeting in London last month, *Microsoft* found itself in the embarrassing position of having to admit that they had inadvertently distributed a virus. The company handed out *Windows 95* demonstration disks to some 160 developers, one of whom took the precaution of running his disk against a virus scanner before use. The Form virus was found, and *Microsoft* was informed. As soon as the alert was raised, *Microsoft* issued a warning to all those concerned.

Dilit Mistry, Marketing Manager for *Microsoft's* Developer Division, said: 'We are naturally running a full-scale investigation onto the incident. Obviously something has gone slightly wrong somewhere; where, we are not yet quite sure. There are contractual obligations which our suppliers must fulfil, one of which is to scan disks before they are sent to us. It may be that the slip-up was somewhere in here; we are simply not sure. Our legal team is working together with the investigation team, and when we receive their report next week, any necessary action will be taken.'

Virus Prevalence Table - January 1995

Virus	Incidents	(%) Reports
Form	20	23.5%
AntiEXE	10	11.8%
AntiCMOS	7	8.2%
Parity_Boot.A	7	8.2%
JackRipper	5	5.9%
Monkey.A	4	4.7%
Cascade	3	3.5%
Jumper	3	3.5%
Natas	3	3.5%
Spanish_Telecom	3	3.5%
Angelina	2	2.4%
Monkey.B	2	2.4%
Ontario	2	2.4%
Tequila	2	2.4%
V-Sign	2	2.4%
AMSE	1	1.2%
B1	1	1.2%
Dinamo	1	1.2%
Diskwasher	1	1.2%
Green_Caterpillar	1	1.2%
Keypress-1216	1	1.2%
PS-MPC-5	1	1.2%
Sayha	1	1.2%
Stoned.Standard	1	1.2%
Wonka	1	1.2%
Total	85	100.0%

This entire incident is extremely embarrassing for *Microsoft*, who, one hopes, really should know better. [*Shouldn't the addition of anti-virus software in MS-DOS 6 make this occurrence a thing of the past? Ed.*] More seriously, the mistake acts as a timely reminder to companies of the need to scan *all* incoming media, regardless of source ∎

Electronic *VB*?

Virus Bulletin is currently considering distributing the journal electronically. However, before any plans are put into action, *VB* would like to assess the demand for such a publication. Additionally, there are several different ways in which an 'electronic' *VB* could be compiled. It would therefore be extremely helpful if those readers interested in an electronic copy of the magazine could contact the Subscriptions Manager, Victoria Lammer, so that any new product can be tailored to suit their needs ∎

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21 February 1995. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

C	Infects COM files	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D	Infects DOS Boot Sector (logical sector 0 on disk)	**N**	Not memory-resident
E	Infects EXE files	**P**	Companion virus
L	Link virus	**R**	Memory-resident after infection

4On	**ER:** A 1346-byte polymorphic virus. No simple search pattern is possible.
Anti_Pascal_II.400.B	**CN:** A minor variant, detected with the Anti_Pascal_2 pattern.
ARCV.Ice-9.639.B	**CR:** A minor variant, detected with the Ice-9 pattern.
Attitude.827	**CN:** Detected with the pattern provided in December 1994 for the 734- and 825-byte variants.
Burger	**CN:** The new variants of this primitive, overwriting virus, 560.AW, 560.AX and 560.AY, are all detected with the Burger pattern.
Cascade.1704.AB	**CR:** A minor variant, detected with the Cascade(1) pattern.
Cholera	**CER:** The two variants, A and B, are both 1497 bytes long. The first contains the text ‘Cholera v1.0 by dr Hellraiser 94-02-03’. The text in B reads ‘Cholera v2.0 by dr Fleischman 93-12-29’. There also seem to be minor code differences.
	`Cholera          5053 5152 5657 1E06 9C3D FEFE 7408 3D00 4B74 11EB 3190 9D07`
CPXK.B	**CN:** A minor variant, detected with the CPXK pattern.
Dark_Avenger	**CER:** This month’s variants are 1800.N, detected with the DA-related pattern, and 2000.Satan, detected with the Eddie-2 pattern.
Grazie	**ER:** The Grazie family used to be called Cossiga, but this has changed after protests from Italy. The two new members are detected with existing patterns: 859 (detected with the Cossiga pattern) and 1361.C, a minor variant detected with the Friends pattern.
Grog	Although many Grog viruses were described last July, there are quite a few left. They are not terribly interesting; sometimes modifications of older viruses, sometimes written from scratch. They share very little code, and almost the only thing they have in common is the author, and (usually) the word ‘Grog’.
Grog.456	**CN:** A simple overwriting virus, also known as Grog.Mormorio.
	`Grog.456         B802 3D47 524F 47BA 9E00 4752 4F47 CD21 4752 4F47 9347 524F`
Grog.495	**CN:** Also known as Grog.Char2Grog.
	`Grog.495         B802 3D90 CD21 9093 90B4 3F90 0690 1F90 BAEF 0190 B9FF FF90`
Grog.518	**CN:** An encrypted virus, also known as Grog.Outwit-C.
	`Grog.518         EB14 908B F48B 34B9 E901 8004 ??80 2C?? E201 C346 EBF4 E8EA`
Grog.557	**CN:** A simple overwriting virus, also known as Grog.Mila.
	`Grog.557         B802 3D2E 8B16 DC02 CD21 8BD8 53B4 3FB9 0300 BAD9 02CD 212E`
Grog.566	**CN:** Also known as Grog.La_Traviata.
	`Grog.566         B802 3DCD 2172 B493 B904 008D 9604 01B4 3FCD 213E 80BE 0401`
Grog.666	**CER:** Also known as Grog.Lor.
	`Grog.666         B802 3DCD 21BB 0057 9372 20CD 218A C180 C91F 4932 C174 1052`
Grog.757	**CN:** Also known as Grog.Crackers.Dream_Team.
	`Grog.757         B800 3D8D 9616 04CD 2193 53B8 2012 CD2F EB12 4707 7207 6F07`
Grog.765	**CR:** Also known as Grog.Miscuglio.
	`Grog.765         B852 3DCD 213C 5374 2AE9 6D01 2121 204D 4953 4355 474C 494F`

Grog.774 **CN:** Also known as Grog.Crackers.Inc.

```
Grog.774        B929 00AC 3CFF 7403 ABEB 0247 47E2 F459 4646 E2E3 B801 B88E
```

Grog.794 **CN:** Also known as Grog.Gsav.

```
Grog.794        B802 3DCD 218B D80E 1F72 3AB4 3FB9 0A00 8BD5 8BFA 83C2 0A8B
```

Grog.902 **CR:** Also known as Grog.Gonfie.

```
Grog.902        B802 3DE8 AB00 7303 E994 0093 B800 57E8 9F00 2E89 160B 012E
```

Grog.903 **CN:** Also known as Grog.Golf.

```
Grog.903        817C 0583 FC75 2281 7C07 FA74 751B 817C 0A83 FC75 1481 7C0C
```

Grog.926 **CN:** This is a polymorphic virus, also known as Grog.Mi_Ami. No simple search pattern is possible.

Grog.990 **CR:** This encrypted variant is also known as Grog.3_0. The search string is the decryptor, and should be used with care as it is rather short.

```
Grog.990        B9D0 03BE 0E01 AC04 ??88 44FF E2F8
```

Grog.1007 **ER:** Also known as Grog.Il_Cuoco.

```
Grog.1007       B802 3D0E 1FBA 2004 CD21 7305 E9EE 0059 C38B E893 B902 00BA
```

Grog.1013 **EN:** Also known as Grog.Sway.

```
Grog.1013       33D2 B440 CD21 51B9 F502 BA00 01B4 40CD 2159 5803 C13B C173
```

Grog.1016 **CN:** Also known as Grog.Crackers.NTA.

```
Grog.1016       8BF8 8DB6 4E04 B912 00F3 A459 E2B6 0E1F BA01 B88E C28D B6FB
```

Grog.1059 **CN:** Also known as Grog.Dan_Zerino.

```
Grog.1059       B802 3DCD 2173 0BE9 8900 3E3E 372F 3933 3C3C BB00 5793 CD21
```

Grog.1142 **EN:** Encrypted virus, also known as Grog.Outwit-E.

```
Grog.1142       161F EB16 908B F48B 34B9 1903 0E1F 8004 ??80 2C?? E201 C3
```

Grog.1146 **CER:** Also known as Grog.Helen. It infects EXE files properly, but COM files will be damaged, and cannot be disinfected.

```
Grog.1146       8935 B440 CD21 72F2 2BC8 75ED B800 4251 5ACD 2172 E58B 4518
```

Grog.1200 **CR:** Also known as Grog.3_1. It uses slightly variable encryption. No reliable search string is available.

Grog.1207 **CN:** An overwriting virus, also known as Grog.Ildono.

```
Grog.1207       A007 01BB D701 3C00 740F 4730 0743 4702 C747 81FB B705 907E
```

Grog.1372 **CR:** Encrypted virus, also known as Grog.2_02.

```
Grog.1372       BE01 01AD 50B9 120F AC8A D05E 81C6 1E01 AC02 C288 44FF FEC2
```

Grog.1641 **EN:** Also known as Grog.Dieta.

```
Grog.1641       B802 3DCD 215A 730B E948 023E 3E34 2F39 333C 3C93 5233 C933
```

Grog.2075 **CER:** Encrypted virus, also known as Grog.4_0.

```
Grog.2075       8BF7 AD50 AD50 83EE 021E 560E 1FBE 4E01 B832 01AB 8CC8 AB52
```

Grog.Enmity **EN, ER:** A group of three overwriting, 512-byte viruses. The first two are not memory-resident.

```
Grog.Enmity.1_0 B802 3DCD 2173 03E9 9900 93B8 0057 CD21 5152 E801 004D 5A8B
Grog.Enmity.2_0 B802 3DE8 AC00 7303 EB7B 9093 0E0E 1F07 B800 57E8 9C00 5152
Grog.Enmity.2_2 80FC 4B74 D080 FC3D 74CB 80FC 5674 C680 FC43 74C1 80FC 4174
```

HLL **CN, CEN, EN:** This month the following C/Pascal viruses have appeared: 3677.B (EN, LZEXE-packed), Linda (CN, 7128), Rust (CEN, also known as Vova.12560) and RSW (5886, CN).

HLLC **P:** There are now two variable-size HLL companion viruses, 8736 and Enrico (which also seems to function as a ‘dropper’ for a Michelangelo-related virus).

Infector.749 **CN:** Detected with the Infector.822 (originally _822) pattern. Five other variants require new patterns:

```
Infector.692.B  A200 01A0 BF02 2EA2 0101 A0C0 022E A202 01B9 0001 BB00 002E
Infector.719    A200 01A0 4A03 2EA2 0101 A04B 032E A202 01B9 9000 BB00 002E
Infector.731    A202 01B9 0001 BB00 002E 8A07 8887 CE02 43E2 F6BA AE02 B920
Infector.765    A200 01A0 0803 2EA2 0101 A009 032E A202 01B9 0001 BB00 002E
Infector.846    A202 01B9 0001 33DB 2E8A 0788 87B0 FE43 E2F6 BA0C 03B9 2000
```

IVP **CEN, CN, EN:** New IVP-generated viruses are: 200 (CN, overwriting), 374 (CEN, overwriting), 478 (CN, overwriting), 665 (EN), 734 (CEN), 766 (EN), 803 (CEN), 811 (CEN), 827 (CEN), 874 (CEN), 886 (CEN), 927 (CEN), 939 (CEN), 974 (CEN) and 2316 (CEN).

Jerusalem.Fu-Manchu.C **CER:** A minor variant, detected with the Fu-Manchu pattern.

Jerusalem.PSQR.D **CER:** A minor variant, detected with the PSQR pattern.

Keycap.Gif.681 — **P:** Detected with the Keycap.685 pattern.

Keypress — **CER:** Three new variants, detected with existing patterns: 1232.O, 1232.P (detected with the Keypress pattern) and 1266 (detected with the Keypress-Mubark pattern).

Leprosy.Skism.808.E — **CN:** A minor variant, detected with the Rythem.808 pattern. Several new variants require new patterns:

```
Leprosy.Skism.1992.C    E803 00E9 F905 51BB 3801 8A2F 322E 0301 882F 4381 FB00 097E
Leprosy.Skism.1818      90E8 0400 90E9 F400 9051 90BB 3D01 8A2F 9032 2E03 0188 2F43
Leprosy.Seneca.483      BB2F 01B9 B601 8A27 80F4 FF88 2743 E2F6 C38B 1EE3 0253 E8E7
Leprosy.Anarchy.469     B927 00BA E101 B44E CD21 3D12 0074 14E8 E000 8B1E F001 535B
Leprosy.5370.A          8B1E 5F01 53E8 1300 905B B9FA 1490 BA00 01B4 4090 CD21 E802
Leprosy.5370.B          8B1E 5E01 53E8 1300 905B B9FA 1490 BA00 01B4 4090 CD21 E802
Leprosy.AOD             E81A 00E9 ED00 8B1E F001 53E8 0F00 5BB9 2B02 BA00 01B4 40CD
```

Lesson_I — **CN:** There are two new variants of this virus, 300 and 306 bytes long. A third 305-byte variant does not work properly, and will only infect 100-byte files correctly.

```
Lesson_I.300            B802 3D8D 944A 01CD 2172 4C93 B43F B904 00BA 2401 03D6 CD21
Lesson_I.306            B802 3D8D 9450 01CD 2172 4C93 B43F B904 00BA 2A01 03D6 CD21
```

Milan.WWT.125.D — **CN:** A minor variant, detected with the WWT-2 pattern. Other new members of the Milan family require new search patterns: AntiNazi (1091), Naziskin.270, Naziskin.335, Naziskin.903, Sabrina (175), Verbatim (289) and Vivisex (683).

```
Milan.AntiNazi          B802 3DCD 2172 C98B D8B8 0057 CD21 8916 0901 890E 0B0 BA00
Milan.Naziskin.270      B802 3DCD 2172 CC8B D8B8 0057 CD21 8916 0701 890E 0901 BA00
Milan.Naziskin.335      BB15 01B9 3A01 2E8A 1780 F2?? 2E88 1743 E2F4 EB01 90
Milan.Naziskin.903      B802 3DCD 2172 C98B D8B8 0057 CD21 8916 0901 890E 0B01 BA00
Milan.Sabrina           B802 3DCD 2172 C48B D8B8 0057 CD21 8916 0901 890E 0B01 BA00
Milan.Verbatim          B802 3D90 CD21 9072 BC90 8BD8 90B8 0057 90CD 2190 8916 0901
Milan.Vivisex           B802 3DCD 2172 BF8B D8B8 0057 CD21 9090 9090 9090 9090 9090
```

NRLG — **CR:** This new virus-creation tool, named NRLG (NUKE's Randomic Life Generator), generates viruses which are not very difficult to detect. As PS-MPC viruses, no search patterns are provided. Users are advised to verify that their anti-virus tools will detect NRLG-generated viruses. Currently the following variants exist: 755, 824, 855, 865, 901, 964, 985, 1007 and 1009. More are undoubtedly on the way.

PCBB.3072 — **CER:** There are two variants of this virus, A and B. Both are encrypted and no single search pattern will detect all instances of the virus. What is interesting, however, is that the encryption method depends on the day of the week, so there are basically only seven different decryptors for each variant.

Pixel.850.B — Detected with the Pixel.936 pattern.

Poison — **CER:** A 2416 byte virus, which uses variable encryption. No simple search pattern is possible.

PS-MPC — **CEN, CER, CN, CR, EN, ER:** The arrival of a number of new PS-MPC-generated viruses should not surprise anybody. This time we have the following: 311.B (ER), 388 (CN), 480 (CER), 504 (CER), 517 (CN), 564.C (EN), 564.D (EN), 565.I (CER), 578.N (EN), 578.O (EN), 578.P (EN), 78.Q (EN), 598.D (CEN), 606.G (CEN), G2.Puppet (CR, 478), G2.Stargate (CER, 525), HD (CN, 497), Mercenary (CER, 854), Page.780 (CN), Payrise.874 (CER), Shrimp (CN, 377) and Snort (CR, 405).

Syslock.Syslock.F — **ER:** A minor variant, detected with the Syslock pattern.

Trivial.46.B — **CN:** A minor variant, detected with the Minimal-46 pattern. Several other variants require new patterns, including the 1600-byte LSD, and a 22-byte virus, which is the smallest known.

```
Trivial.22              2A2E 2A00 B44E 8BD6 CD21 B43C BA9E
Trivial.29.E            1801 CD21 B802 3DBA 9E00 CD21 93B4 4083 C262 CD21 C32A 2E43
Trivial.33.B            3DBA 9E00 CD21 8BD8 B921 00BA 0001 B440 CD21 CD20 2A2E 2A00
Trivial.85              CD21 5152 B440 BA00 01B9 5500 CD21 B801 575A 59CD 21B4 3ECD
Trivial.90              CD21 73E3 B409 BA36 01CD 21CD 2063 3A5C 2A2E 636F 6D00 202D
Trivial.92              B44F CD21 73E3 B409 BA35 01CD 21CD 202A 2E43 4F4D 0055 6E64
Trivial.LSD             CD21 8BF3 B801 4333 C98D 541E CD21 B802 3DCD 2193 B440 B940
```

Vbasic.G — **EN:** A minor variant, detected with the Vbasic (5120) pattern. It seems damaged, and during testing it was not able to infect COM files as the other variants do.

VCL — **CN:** The following overwriting variants appeared this month: 457, 1297, Fire, Mindless.423.C, Mindless.423.D, Mindless.423.E and Viral_Messiah.705. The following 'disinfectable' variants also exist: 342, 511, Anston.B, Bev.516.B, Code_Zero.652.B, Genesis.738, Genocide.952 and Pleasure.512.

Vienna — **CN:** There are now three Vienna variants: 620 and 648.AF (detected with the Vienna-2 pattern) and Violator.5305 (detected with the Xmas-Violator pattern).

INSIGHT

Caught in the Crossfire?

Anyone who has followed the virus problem over the past five years cannot have missed the name of Sarah Gordon. Well known in the electronic community for her work in ethics and technology, she has been a constant voice of reason and moderation. This has given her a unique position, being trusted by many virus writers and anti-virus software developers.

However, her achievements are confined neither to the *Internet* nor to viruses. Her research projects at Indiana University include *UNIX* security and Computer Ethics. She is a regular contributor to *UNIX* security journals, and her work in ethics and technology was awarded the best paper/presentation distinction by *Sec 94 IFIP TC 11*. She does not voice her opinions lightly: this is the first interview she has given on her views of the virus and security world.

The Virus Adventure

Gordon's recounting of her first experience with a computer virus recalls a familiar story - her machine became infected. 'I got this old *XT*, and it was really slow and it kept locking up. I thought that something was wrong, and as I had been reading the Virus echo on *Fidonet* for a little bit, I realised that I might have a virus. I downloaded a copy of *McAfee Scan*, and sure enough, I had PingPong.B. I followed the instructions, typed CLEAN, and it was gone. I thought that was great... and the next day it was back again.'

At the time, she had problems convincing those around her that the problem was real; in 1991, viruses were still very much a novelty. 'Nobody believed that I had a virus; they would say that I was just making it up, and that *nobody* got viruses. I was really upset, because this virus just kept coming back, so I wrote to a vendor, "I think I have this terrible new virus, and it won't go away. I would send you a sample, but I don't know how...".'

Gordon then scanned every file on every disk she owned, including those for her *Tandy CoCo*, unzipping every ZIP file - even though PingPong is a boot sector virus. 'I wasted about six weeks before someone on the Virus echo told me "Here. This is what you really need to do". That was the start of my virus adventure, I guess.'

The 'adventure' continued in the *Fidonet* Virus echo. At the time, the Sofia virus exchange BBS was a hot topic. 'Everyone was talking about viruses, and a new virus exchange BBS in Sofia. I tried to call it a couple of times, but never got through. Many of the conversations centred around how the virus writers were bad... it was very much the good guys against the bad guys - really polarised.'

Gordon has good memories of the time spent on the echo: 'It was really interesting. I was meeting all these new people and learning a lot. And then he came.'

He, of course, was the Dark Avenger.

<dav>, MtE and All That...

From the first time Gordon saw the Dark Avenger's messages, she was fascinated. 'People would argue and argue, and he would post just one line. I just felt like I knew him from the first time I saw his comments.'

This was the start of a relationship which has linked Dark Avenger's name inextricably with her own. 'I don't know if you remember or not, but at the *EICAR* conference, I just had a slide with his name on it. I said, "Here's the subject of the Dark Avenger. Everyone expects me to talk about him. So I did." That's really all I can say. It *has* been a long time. It was 1992 when he last released a virus - that's almost three years ago.'

Even though the Dark Avenger has been not been active for a long time, the press is still fascinated by him, although Gordon is at a loss to explain why. 'Maybe he was an innovative programmer; maybe he had new ideas. But a lot of people have new ideas. Why is anyone famous? Right place at the right time. Wrong place at the right time. As a person, I don't think he's any worse than a lot of people.'

The reasons for his infamy may be less than clear, but famous he is, and the electronic friendship which developed between the Dark Avenger and Gordon still intrigues many people. Gordon is by now accustomed to questions on the subject, and welcomed the chance to give her side of the story. 'What attracted me to him was not the fact that he was a virus writer, it was the individual. It was the very unique person that he was that I think very few people saw. Some people say that I'm encouraging people to write viruses by talking about Dark Avenger. No - he stopped. The viruses were the thing about him I found least interesting.'

Just before the Dark Avenger stopped distributing his viruses, he consented to an interview with Gordon - this was viewed by some as exploitation, but she is adamant that this was not the case. 'He wanted to say those things and I was the only person he was comfortable talking with. People think I got really rich from this one, but it cost me more in charges for connect time and phone bills than any money I ever made.'

Motivation

Many people attribute virus writing's popularity to the hype surrounding such topics as the Dark Avenger or the Michelangelo virus. Does this not give virus writers the publicity they seem to crave? Gordon disagrees: ‘I think that assumes that a virus writer wants lots and lots of attention. Just like anyone else, some might want attention, some might not. Some may want to see their virus in a scanner, some may not. It’s really a very individual thing.’

Gordon presented a paper at *VB 94* which examined these issues. After completing her research, she found no common factor which motivates virus writers: sometimes the reason was simply to see if it was possible.

‘You’ve been writing this little program which displays “Hello World”,’ she explained, ‘And you suddenly discover that you can write a program which will continue to function even in your absence, and perform some action when you are not there. I know when I was a kid, I used to play around building Robots and taking apart telephones. The fact that things could operate in my absence on my behalf was great. I don’t think that viruses are such a big step away.’

Distribution, Ethics and the Law

One of the biggest problems the community faces is dealing with virus writing and distribution. Gordon believes there is a clear distinction between the two. ‘I think it’s impossible to say that someone isn’t writing viruses... what you do in your own private environment is something that is not usually subject to public scrutiny, and neither should it be. I think that people need to be very careful to distinguish between virus writing and virus distribution.’

However, rules need to be constructed carefully, as it would be all too easy to prevent legitimate actions. ‘There are ways of distributing viruses which I don’t think are unethical at all. People who are working in research groups need to share the virus information among themselves in a controlled manner. That’s virus distribution, but it’s very different from distribution in an uncontrolled manner.’

Then shouldn’t uncontrolled distribution be made illegal? ‘Let me draw an analogy. If I manufacture weapons and I sell them in a responsible manner, it is likely that I can’t be held responsible if someone abuses them. On the other hand, if I manufacture weapons and I sell them in an irresponsible manner, maybe I’m going to be responsible legally, maybe not... but ethically, I’m certainly responsible.’

One of the most frequently-encountered arguments against prohibiting virus distribution is that viruses are a form of free speech, and therefore cannot be curbed. This subject is of special interest to Gordon, who has studied constitutional law and free speech issues.

‘I’m concerned that people equate writing viruses with free speech. At least in America, free speech was never intended to cover that particular type of activity; it was designed to provide a form of redress against the government.’

‘When I see someone saying that viruses are a “right”, I really have to question that,’ she continued. ‘I don’t think that it’s ever been established in a court of law that anyone has the right to distribute damaging executable code. As far as source code goes... maybe it is, maybe it isn’t. I don’t even think that’s really the issue. We have a lot of things which are rights, but it doesn’t mean that they are ethically correct things to do, or desirable things.’

‘When we look at virus distribution, we need to ask ourselves if this is the right thing to do. Is this something we really want to do? If everyone did this, what would be the result? I think this is a really large question, and we haven’t examined it completely yet. We need to do that, especially before we make any decisions about making laws.’

The Business World

Although much of Gordon’s work is concerned with ethics, that is only one facet of her studies. Viruses cause damage, and the situation must change. ‘Viruses are definitely hurting businesses and so we have to protect against them,’ she agreed. ‘We can’t ignore that, because they are a real problem, causing real dollar damage. Real people get them. I talk to students in small colleges that have got viruses. I talk to individual users - viruses. I talk to big organisations - viruses. Universities - viruses. I don’t want to sound like “Viruses everywhere!”, but they’re a real problem. It’s kind of a mess.’

“If it is sides, then yes, I am caught in the middle. If that’s what it takes, then so be it”

Is this partly the fault of the industry? - if it were doing its job, computers would be protected. Gordon thinks not: ‘When you have a software product, no matter what it is, users have some responsibility. People who make desktop publishing software are not responsible if the publication looks bad. However, we do have certain responsibilities as an industry: we need to supply good and accurate information, detect viruses and give good service. If a company provides all of that, then it is fulfilling its obligation. Not all companies do that. Will I name them? No.’ But could she? ‘Yes!’ she laughed, refusing to be drawn further.

One company which Gordon obviously feels is fulfilling its obligations is *Command Software Systems*: after many years of being unaffiliated with any anti-virus vendor, Gordon has chosen to move to Florida to work for the company. Why?

‘Well, it’s a great product! I ran a BBS for a long time, and *F-PROT* was always one of the most popular scanners. During the big Michelangelo scare, I spent a lot of money to buy diskettes to send the program out to people who were worried and asked me for help. Why *F-PROT Professional*? Actually, it’s really “Why *Command*?” They make a commercial version, which offers even more features. I think it’s important that commercial users have a company which provides support. I’ve watched the kind of support *Command Software* provides and it’s excellent.’

This is Gordon's first affiliation with an anti-virus company. How does it feel? 'Take a look around you. It's like a dream come true. People always used to ask me what would I like most: I would always say "Hang out on the *Internet*, work with *F-PROT* and live in Florida".'

The Personal Touch

Perhaps one of Gordon's most admirable achievements is the fact that she is respected and trusted by both 'sides' of the battle. How has she managed to do it? 'There's something which you say which makes things very clear. You say both "sides" trust you. I don't see it as a two-sided thing. These are people who are doing different things. If it is sides, then yes, I am caught in the middle. If that's what it takes, then so be it.

'I don't know if you remember how virus writers and anti-virus people treated one another in public forums, but for the most part there was a lot of disrespect and name calling. Nothing was being accomplished; people were getting more and more provoked and frustrated. Now, at least more people are talking to each other.'

The problem which exists between virus writers and developers stems in part from the way computers can remove normal values from social interaction: 'If I walked up to you in the street and slapped you in the face, people would say, "Wow, what's that about?", and you would rightly say, "What are you doing?". If I do the same thing electronically, people just look the other way. Maybe we've somehow forgotten that they are all still people. We lose that in technology.'

Hacking and Other Issues

Viruses are only part of Gordon's involvement in computers: she is also known for her system penetration skills, and has acted as a consultant in securing systems. Would she call herself a hacker?

'I guess you could say I come from the old school,' she mused, 'when hacking was not a dirty word. Do I have friends who are hackers? I have friends who are considered hackers, some who consider themselves hackers, and some who don't even know what a hacker is.

'If you define hacking as an illegal act which compromises a system, then of course I would not be a "hacker". But if you define it as having the skills to push a system to its limits, then sure, you could call me a hacker I guess. But don't you think labels like this tend to mislead people?'

Does she have direct links with any of the hacking groups? 'I spoke at *Defcon*. It's a "hackers' conference", and I spoke about privacy and anonymity. Padgett Peterson spoke about E-mail, Phil Zimmerman spoke on the panel I organised. No, we didn't use names like "Digital Jihadster", but we were there. Actually I go by the name "Theora" - from the *Max Headroom* TV series - when I'm Net-cruising.'

'I have been invited back to *Defcon* and will be speaking there this coming August,' she continued. 'I believe in communication. *Defcon* is a chance to talk about controversial subjects in an open forum. Usually people are receptive to discussion with people from "the other side". I mean, with the exception of my being called homely after the conference, I didn't get any negative feedback from anyone. Most of my friends attend *Defcon* and similar conferences. I don't think there is any big controversy there.'

The Road Ahead

Her job with *Command* will inevitably change much in Gordon's life; however, she is committed to her current research. 'I plan to continue doing research into ethics and technology - something which *Command* is very supportive of. I still plan to do work with *UNIX* security, which is my first love. I just want to make good information available, so people can assess their security for themselves.'

"I don't want to sound like 'Viruses everywhere!', but they're a real problem. It's kind of a mess"

But what of the future for the industry? 'New platforms. New viruses. New threats. It will require a little more talent to do, so I think it will not develop at the same rate it did for DOS. It was simple to write a virus under DOS. People have always said that viruses are so exciting. But how many Find First, Find Nexts can you stand?'

'Writing *Windows* viruses is much more difficult, and if you have that talent, you are probably not going to waste it writing viruses. How many really talented virus programmers have there been? Very few, thankfully. There may be more impact, because people might not be expecting them.'

Gordon believes that the way forwards is more about people than technology. 'I think the long-term solution is user education, and the implementation of an ethical curriculum at a very early age, so that we stop writing viruses, and start taking the security of data a little more seriously.

'They need to take security and integrity seriously, and not take it for granted. Nobody will reach down out of the sky and secure a computer. Companies are going to have to do it for themselves. They need to be informed, and make good decisions, and make sure that they educate their employees, so that they use these products correctly and consistently.'

'There's a poster which says, "This computer is protected by a false sense of security", and that's what I see in a lot of places.' Will that change with time?

'I certainly hope so; otherwise my work, and the work of a lot of other people, will have been for nothing. I want to see a time when we all work together towards the same goal.'

FEATURE

Viruses on Windows NT

Ian Whalley

The question of the susceptibility of *Windows NT* to viruses is one which is being raised increasingly frequently, as companies begin to switch over to *Microsoft's* new operating system. However, until now, there have been (to my knowledge) no documented tests of viruses on systems running *Windows NT*. To correct this, I carried out a series of tests, the results of which form the basis of this article.

NT: A Brief History

Windows NT is *Microsoft's* high-end product, designed for processor-intensive applications and networking. Unlike their previous products bearing the *Windows* logo, it is specifically designed with networking in mind. This is in accordance with the widespread belief that in the 1990s, computer users will be even more keen to interconnect their machines, by utilising the peer-to-peer or client-to-server architecture supported by *NT*.

There are currently two operating systems in the 32-bit marketplace to which *MS-DOS* users might move: *Windows NT* and *IBM's OS/2 Warp*. *UNIX*, with its idiosyncrasies and multitude of mutually incompatible versions, is not a realistic possibility for most applications.

The attractions of *Windows NT* are many: it has the weight of *Microsoft* behind it, it supports a wide variety of networks, and it can run on several platforms (*IBM PCs*, *DEC Alpha* and *MIPS* are all supported). However, its biggest disadvantage is the high machine specification required. Minimum configuration is a 386/20 with 10MB RAM and an 80MB hard disk, but even with this it is barely possible to do anything other than boot up and say, 'Oh look, it looks just like *Windows*'.

Running applications on such a system is not a practical option. No *NT* developer would consider working on anything less than a 486/66 with 20MB RAM and a 300MB hard disk. Having said that, *NT* is gaining in popularity, especially with the pull of *Windows 95*, which is tailor-made to cohabit with *NT* in Bill Gates' vision of a radiant future networked by *Microsoft.*

Part 1: The Current Threat

Putting the Boot In

When a PC boots, the initial stages of that boot are independent of the operating system (OS) which the PC will run. Clearly, the hardware has no prior knowledge of this OS. If a user attempts to boot a PC with an infected diskette in the A: drive, that virus will infect the hard disk. With this in mind, I carried out the tests described below, using a mimimally-configured machine.

When *NT* is installed on a PC from DOS (i.e. the user boots DOS, accesses the CD-ROM, and proceeds with *NT* installation), the machine becomes 'dual-boot'. When the PC boots from then on, the user may select *NT* or DOS as the operating system for that session.

During installation, *Windows NT* takes a copy of the original partition boot sector (the first sector of the active partition, which contains a copy of DOS), storing it in the root directory of that partition as the file 'BOOTSECT.DOS'. I have had one report of a machine infected with Form having *Windows NT* installed: in this case, the virus is present in 'BOOTSECT.DOS' after installation is complete, and will become active if DOS is selected from the boot menu.

Forming an Opinion

When I infected a previously clean *Windows NT* machine with Form, everything booted up well. The virus was in the partition boot sector, and the original code in the expected position at the end of the physical disk. However, after using the machine for a while, and rebooting a couple of times, further access was not possible.

The original copy of the boot sector had been overwritten by the system paging file during my last session; consequently, when the virus directed execution to that sector, it did not find a valid partition boot sector, and the machine hung. I encountered this problem at an early stage because the hard disk in the machine was barely large enough to support *NT*, so the paging file was using almost every sector on the disk not taken up with the operating system.

Booting from the *Windows NT* set-up disks, selecting the 'repair a damaged installation' option, and deselecting all subsequent options with the exception of 'examine boot sectors' enabled operations to continue. This worked whether or not I gave it the emergency repair disk. There is in this case no need to resort to a virus scanner to remove the virus - the set-up disks can do it for you.

Master Boot Sector Viruses

When planning this article, I intended to give detailed information on what happens to *NT* when the machine is infected with various Master Boot Sector (MBS) viruses, and advice on what to do for each (see Figure 1 for a list of viruses used for testing purposes, and the results). The advice, it transpired, is simple. If your *NT* machine becomes infected with a MBS virus, and you have not taken adequate precautions, you may have great difficulty in restoring your system. The reason for this, which at first sight may appear apocalyptic, is that for most viruses tested, *NT* fails to boot. Everything appears normal until the screen changes colour and *NT* begins internally to map logical partitions on the hard disks to internal device names. Then the system hangs with a 'blue-screen error'.

This means that the kernel has encountered a fatal error and cannot continue. I have only once before seen an error of this type under *Windows NT*; the culprit then was an incorrectly-configured SCSI card. A 'blue-screen error' can record one of two things: either INACCESSIBLE_BOOT_DEVICE, or 'This system may be infected with a virus'. Whilst the second is marginally more helpful, the effect of both is the same - you cannot boot *NT*.

To make matters worse, attempting to fix the problem with the set-up disks was not successful, even with the aid of the emergency repair disk. This is surprising: why should the repair facility only look (as seems to be the case) at the partition boot sectors, and not at the most frequently attacked parts of the PC, the MBS?

The distinction between viruses which allow *NT* to boot and those which do not is currently unclear. Differences may lie in which areas of the MBS are overwritten, or the manner in which they hook interrupts. Whatever the reason, if the machine boots successfully, I have been unable to replicate the virus with which it is infected. This is unsurprising: the viruses will be relying on knowledge of how DOS loads, and how it subsequently manages the interrupts. *NT* breaks all these 'rules', rendering the virus impotent. In this case, the user will probably not realise he has a problem.

What to Do

So what can you do if your *NT* machine becomes infected with a boot sector virus which prevents the system booting? Here, *Windows NT* comes to our rescue to some extent.

Following discussions with a user in the US, and with *Microsoft*, some interesting things about *Windows NT* systems have come to light. Firstly, if your partition boot sector has been corrupted on an NTFS partition (New Technology Filing System - the more efficient *Windows NT* replacement for DOS FAT and *OS/2* HPFS), it is possible to retrieve a copy of the original of that sector.

When *NT* formats an NTFS partition, it copies the first logical sector (partition boot sector) to the exact centre of the partition. This can be retrieved using a DOS utility such as *Norton Disk Editor*, and copied over the infected one. It is necessary to boot DOS to do this, but that is to be expected, due to the lack of *NT* disk utilities.

This works because current DOS boot sector viruses are not aware of this copy, so they cannot corrupt it. If *Windows NT* leads to partition boot sector viruses of its own, they may well take steps to alter the copy as well as the original.

Windows NT, in addition to the copy of the partition boot sector, may have a copy of the MBS. I do not know where this copy originates, but on every such machine I have seen, one has been stored *somewhere* on the disk. Interestingly, it is not necessarily aligned with a sector boundary, so it does not appear to have been placed there by a sector-to-sector copy. It is possible that it is part of the secure area of the registry. Consequently, if all else fails, the disk can be searched for the MBS copy, and the corrupted MBS replaced with that.

This uncovers another seemingly undocumented feature of *Windows NT* - the fact that it is possible to create a boot diskette which is sufficient to load *NT* without executing either of the boot sectors on the fixed disk.

Such a diskette must be formatted within *NT* (its boot sector structure is different from a disk formatted under DOS), and have the following hidden files from the root directory of the active partition copied onto it: NTLDR, NTDETECT.COM, BOOT.INI and, if present, BOOTSECT.DOS.

Using this diskette, it is possible to boot a *Windows NT* machine even when it is infected with a virus. Then you can use a *Windows NT* virus scanner to remove the virus. Of the viruses tested, only the Monkey variants prevented a boot under these conditions - this is because Monkey does not preserve the partition table.

If a system file becomes infected, this diskette will not enable you to avoid executing the infected file. However, it is doubtful whether an executable infected with a DOS virus and used in the *Windows NT* boot process could do any damage. Once *Windows NT* has booted, you can carry out whatever repairs are necessary.

Virus Name	Effect on Windows NT
AntiCMOS	NT Boots okay, no infection of floppies
AntiEXE.A	Blue-screen, 'INACCESSIBLE BOOT DEVICE'
Monkey.A	Blue-screen, 'INACCESSIBLE BOOT DEVICE'
Monkey.B	Blue-screen, 'INACCESSIBLE BOOT DEVICE'
Natas.4744	Blue-screen, 'This system may be infected'
Parity_Boot.A	Blue-screen, 'INACCESSIBLE BOOT DEVICE'
Peanut	Blue-screen, 'This system may be infected'
Spanish_Telecom	Blue-screen, 'INACCESSIBLE BOOT DEVICE'
Stonehenge.B	NT Boots okay, no infection of floppies

Figure 1: Various Master Boot Sector viruses were run in *Windows NT*. This table shows the results of each test.

Part 2: The Future

The Bad News

What of viruses specific to *Windows NT*? In the words of a Pogo cartoon strip, 'We have met the enemy and he is us'. For what have we done? In our search for ever more elaborate methods of using and linking computers in the modern environment, we may have handed virus writers the ‘keys to our kingdom’. Until now, the most difficult task for PC viruses has been to spread not on any given machine, but to other computers. Viruses to date have not been network-aware; they rely on the human factor - people passing disks around, or exchanging executables over networks.

Windows NT offers users the carrot of ‘invisible’ networking. When I click on an icon on my *NT* desktop, that application could be on my hard disk, on the hard disk in the machine across the room, or on the hard disk of a computer I reach by an ISDN link across the country. Networking is also largely invisible to the application: if I instruct a word processor to save my document on drive G, it does not know if that drive is local or remote; nor does it care.

It has thus become easy for viruses to spread across organisations like wildfire. Even current DOS executable file viruses should find it easy to infect remote executables under *NT*, but I have not yet had an opportunity to test that.

“with each process operating in its own protected address space, such viruses will not have the power they have today”

But wait! Imagine a virus, tailored for the brave new world of *Microsoft* networking; a file virus which, when triggered, examines drive mappings set up by the currently logged-in user. When it finds a mapping to a drive share which has been left writeable, it examines files there. Likely candidates are found, and examined to see if they can be modified.

If write access is available to the security identifier under which we are operating, the file is modified in much the same way as DOS viruses, and bingo! a remote machine is infected. As soon as any of those executables is run, the process starts again. Furthermore, it is not even necessary for a drive mapping to exist. Our *Gedankenvirus* could query the network, evaluate remote machines and drive shares available on those machines. Then, using the UNC (Universal Naming Convention), it could gain access to the shared drives, and proceed as described above.

Luckily, *Windows NT* fares better against memory-resident viruses: with each process operating in its own protected address space, such viruses will not have the power they have today. Hooking interrupts in a pure *NT* application is impossible, unless the application is a device driver, in which case it can accomplish much the same thing.

While booting a *Windows NT* machine, countless device drivers are started - one extra would not be noticed. It is not difficult to install a driver which sits just above the level of physical disk drivers, and intercepts reads and writes to and from those devices. Once installed, the question of access privileges does not arise; a kernel-mode driver runs as the system. Byte swapping on random disk writes, surreptitiously presenting the calling application with a different sector from that requested, reducing system performance by spin-locking the kernel for milliseconds, even gradually removing data from disks; all this and more is possible.

The networking aspect of *Windows NT* has the greatest possibilities from the virus author’s point of view. We are moving to an era of desktop computing where the visions of cyberpunk authors could become more real. Since Brunner wrote of his ‘tapeworm’ in 1975, we have had the *Internet* Worm of 1988: with today’s technology, such a beast may become possible on computers to which many have access; those in the office, rather than computers in the ivory tower.

The Slightly Better News

We have handed virus writers access to the virtual equivalent of an Aladdin’s cave - but all is not lost. Virus writers *could* do all this; but they are, in the words of the same Pogo cartoon, ‘surrounded by an insurmountable opportunity’.

Developing for *Windows NT* is not cheap. The average virus writer’s machine is not capable of running *Windows NT*, even if he had access to the software. Whilst 486 PCs are cheap now (with prices dropping all the time), the required amount of memory easily costs more than the remainder of the hardware. *Windows NT* workstation v3.5 comes on 22 installation disks, or on one CD-ROM.

Even after *NT* is installed, the virus writer will come up against the obstacle of how to obtain information. Countless books on DOS exist to tell the reader all he needs to write a virus (undocumented interrupts, etc), but there is a dearth of such knowledge for *NT*. *Windows 95* will doubtless produce a flood of books on its internal workings, many of which are similar to *NT*. However, while information is at the moment at an enormous premium, it will not always be so.

Conclusion

I warn again: be careful with your *NT* machine. Where possible, use the CMOS to disable booting from floppy by default. Creating a clean boot disk is easy, but once within *NT* the only tool available is to attempt to disinfect the specific virus contracted.

A more general, and thus advisable, course is using *Norton Disk Editor* (or something similar) in DOS to copy all boot sectors: keep them safe on a diskette. This should be done after running the disk manager for the first time, as it stamps a value into the MBS the first time it is run. *NT* may become upset if this value is not present on subsequent boots.

If anything dire does happen to your system, try booting DOS from a clean diskette once more and using the same utility to copy the sectors back. This way you can be sure that your critical sectors are back as they should be. The fun really starts when the virus has not preserved a copy of the original MBS. Forewarned is forearmed.

VIRUS ANALYSIS 1

Sampo: Packing a Wllop?

Sampo is a Master Boot Sector (MBS) virus known to be in the wild in Singapore and in the UK. Its manner of infection is identical to other such viruses: when the user attempts to boot from an infected diskette, the virus makes modifications to the MBS, then goes memory-resident on subsequent boots, infecting floppies as they are accessed.

Operation

Just like almost all MBS infectors, Sampo obtains the size of system memory from offset 13h in the ROM BIOS data area, copies the resident part of its code to a point just below the top of memory, and 'reduces' the amount of system memory to hide itself.

It then uses data in the boot sector to locate the five sectors containing the remainder of the virus code on disk: the cylinder and drive number are stored at offsets 40h and 42h within the boot sector. It uses Int 13h function 2h to read the sectors into memory, and executes a jump to that location.

The main body of the virus code installs handlers for interrupts 08h (timer), 09h (keyboard), and 13h (ROM BIOS disk services), directly patching the interrupt vector table. The Int 08h handler is installed when the PC is booted from diskette, or if the date is 30 November; the Int 09h handler on booting from a diskette. The Int 13h handler, however, is a permanent part of the process. Once installation is complete, the original MBS, stored at sector 14 of track 0, is loaded and processing passes to it.

Interrupt 13h Handling

Sampo just about succeeds in being a 'stealth' virus by hiding the changes to the MBS. It does not attempt to hide the other alterations made to track 0, nor does it hook writes to fixed disks. As a result, it is possible to clear an infection using FDISK / MBR without first clean booting, although this is not generally recommended.

However, Sampo does pay attention to attempts to write to floppy boot sectors - there is an inordinate amount of code concerned with manipulating disk writes so as to infect clean floppy diskettes.

Interrupt 09h

This handler appears to be intended to detect the user pressing Ctrl-Alt-S. The virus does this by reading from port 60h (keyboard scan), and checking the value held in the lower of the two keyboard status bytes held in the ROM BIOS Data Area. This has bits two and three set if the Ctrl and Alt keys (respectively) are down.

If the key sequence which generated this particular instance of the interrupt is Ctrl-Alt-S, an elaborate and lengthy trigger sequence (close to 150 bytes) fires. It seems intended to write characters to the screen and to trigger the speaker before rebooting the PC with an Int 19h call. Unfortunately, it did not activate on the test machine - this may be because the virus attempts various types of direct port access, some or all of which may not be supported.

Interrupt 08h

The handler for Int 08h also uses the ROM BIOS Data Area, obtaining the data in byte offset 6Eh, one of the four bytes containing the system time in timer ticks (18.2 per second). It does some calculations on this byte, and attempts direct access to port 3DAh, which is assigned to graphics devices which support CGA or better graphics.

It is not clear exactly what this interrupt handler is attempting to do with these devices; once again I could not make Sampo trigger on the test machine.

Summary

Sampo is long for a boot sector virus, due to the size of its trigger routines. It stores its code in five sectors of track 0 on a disk. Its stealth capabilities are unimpressive, displaying no new techniques. It is a very ordinary virus - if this is the best virus writers can do, there is hope after all.

Sampo

Aliases:	Wllop.
Infection:	MBS on hard drive, floppy boot sector.
Self-recognition on Disk:	
	27 bytes starting from offset 4F in MBS.
Self-recognition in Memory:	
	None.
Hex pattern (in boot sector and in memory):	

```
FA8C  C88E  D88E  D0BC  00F0  8BC8
83C1  06A1  1304  BB00  028B  D025
```

Intercepts:	Int 13h for stealth, Int 08h and Int 09h for trigger routine.
Trigger:	This activates if Ctrl-Alt-S is pressed after booting from floppy, or if the date is 30 November.
Removal:	Under clean system conditions, use the FDISK /MBR command. See text for further details.

VIRUS ANALYSIS 2

In the Dragon's Lair

Eugene Kaspersky
KAMI Associates

Anti-virus research can be a very challenging field: virus authors are continually coming up with new ideas, or bringing two existing ideas together to form a new threat. A typical example of this process is Dragon, a virus which combines the properties of a cavity-infector which behaves like a device driver once memory-resident.

Methods of Replication

A parasitic virus can use one of several different techniques to infect an EXE file. The most commonly encountered method is to append the virus body to the end of the file, and adjust the data stored in the EXE header (such as the initial values of the Code Segment, the Instruction Pointer, the Stack Segment and the Stack Pointer registers) to ensure that control passes to the virus code.

Yet another type of EXE infector searches for unused space in the EXE headers (this infection technique is a special case of that used by cavity viruses). EXE headers often have such space which is padded with zeros, to a maximum length of 480 bytes. This is sufficient for a virus to install itself, and proves an admirable place from which it can spread. The advantage of this technique is that files do not change in length when they are infected.

Viruses belonging to this family often use non-standard manners of infection: instead of hooking the Int 21h handler to intercept and infect EXE files on execution or access, they hook the Int 13h vector controlling disk access and test the sectors accessed for the EXE stamp MZ. If that marker is found, the virus inserts itself, and corrects the EXE header so that control passes to the virus when the file is executed. Should that sector (and correspondingly, that file) already be infected, the virus will typically perform a stealth routine. However, Dragon finds yet another way of operating when it is memory-resident, hooking no interrupts, but still managing to infect other files.

The Virus in Action

Dragon occupies 400 bytes of code and data, writing itself from offsets 0070h - 0200h in the EXE header, and occupying almost all of the EXE header. This is a length which coincides with the sector length of most disks, enabling the virus to fit neatly into the first sector of an executable file.

When the virus infects a file, it adjusts the fields of the EXE header, making its code the first block of the EXE module code. This ensures that the virus receives control on execution. When an infected file is executed, control passes to the virus' installation routine. This allocates 400 bytes at the top of base memory (using standard DOS calls), marks the MCB owner field of that block as 'system' to hide its presence from a casual search, and copies itself there.

Next, the virus obtains the DOS Data Table pointer by using the undocumented Get List Of Lists call (Int 21h, AH=52h), and scans the list of Drive Parameter Blocks for hard and floppy disk drivers.

If such a driver is found, the virus stores the addresses of its Strategy and Interrupt handlers, and sets the address of the virus' TSR code as the address of the original device driver. This done, installation is complete, and the virus returns control to the host program.

After installation, therefore, every call to the device drivers concerned with disk access transfers control to the virus code. This code has the properties of a device driver, complete with a device header, device attribute field, and its own Strategy and Interrupt routines.

Device Driver Routines: Infection

On calls to these drivers, DOS passes control to the virus' Strategy and Interrupt handlers. The virus intercepts and processes the necessary functions, and in turn passes control to the code of the original device drivers.

0000h	EXE file signature 'MZ' or 'ZM' (4D5Ah or 5A4Dh)
0002h	Length of file MOD 512
0004h	Size of file in 512-byte pages
0006h	Number of relocation table items
0008h	Size of header in paragraphs
000Ah	Minimum number of paragraphs needed above the program
000Ch	Maximum number of paragraphs desired above the program
000Eh	Segment displacement of stack
0010h	Initial contents of SP
0012h	Word checksum
0014h	Initial contents of IP
0016h	Segment displacement of Code module
0018h	Offset of first relocation item
001Ah	Overlay number (0 for resident part of the program)
001Bh	Variable reserved space
	Relocation table
	Program code and data

The data stored at the start of an EXE file. The Dragon virus copies its code into the slack space often found in EXE files which do not have a large relocation table.

The virus' Interrupt routine contains only one instruction, RETF (RETURN FAR), which immediately returns control to the program performing that call. All the information necessary to carry out infection is placed in the code of the virus' Strategy routine.

On receiving control, the Strategy routine calls the original driver's Strategy and Interrupt routines to ensure that normal disk access is maintained. It then checks to see whether the disk access has been successfully executed. Next, the virus obtains the command number of the request which has just been completed from the Device Request Header. If this was either a Read From Device, Write To Device, or Write With Verify command, the virus checks the beginning of the data transfer buffer for the MZ stamp. If this is found, the virus chooses between passing control to its infection or its stealth routines.

Where the code of the EXE header from 0070h - 0200h is not filled with zeros, the virus compares 28h bytes of its code with the bytes at offset 0070h in the EXE header, to check whether the file is already infected.

If the virus decides that a file is infected, Dragon performs a stealth routine on Read, restoring all the EXE header fields altered on infection (EXE header size and initial values of registers) and fills the data from offsets 0070h - 0200h with zeros. File length does not grow on infection, nor is the time/date stamp changed, as the virus does not use any DOS calls during the infection routine.

The stealth routine has another benefit: the virus does not check its already loaded TSR copy. This works because when any infected file is next executed, the stealth routine substitutes the infected code with the original, and does not allow the virus installation code to be executed twice.

If the EXE header is filled with zeros at offsets 0070h - 0200h, the file is deemed suitable for infection. The virus then decreases the number of paragraphs in the EXE header, increases the initial stack segment value to avoid stack overlap, makes a copy of the initial values of the EXE entry point, and stores the address of the virus' entry point there. Then the virus copies itself into the slack space in the EXE header, saving it on disk by calling the original disk device driver with the Write To Device command.

The result of all this is that EXE files become infected whenever they are accessed: on execution, on copying, on modification, etc. This infection technique places the virus into the category of 'fast infectors'. Additionally, the virus uses only device driver calls; therefore it infects files with a read-only file attribute, and does not cause a DOS error message when an attempt to write to a write-protected diskette is made.

Conclusion

A surprising discovery on disassembling this unusual virus was that it contains a bug, which appears when the virus infects files with non-standard EXE headers. Usually, small EXE files have a header of 200h bytes. Where this is not the case, the virus will corrupt the file on infection, and store incorrect values in the EXE header fields. When such a file is executed, the system will crash.

The virus does not pay close enough attention to the contents of the EXE header's relocation table. If the table is large and overflows into the area above offset 0070h, the virus will not infect the file, because the area the virus would occupy is not zero-filled. If, however, the relocation table is small (up to 20 entries), the virus may cause corruption of infected files.

Dragon may have problems working correctly under *NetWare* and in a multitasking environment. Although I have not carried out any tests, there may be compatibility problems between techniques used by this virus and such complex software as the *Novell* remote disk drivers and *Windows NT* and *OS/2* operating systems.

In any event, Dragon is a complex and interesting virus, and one of the few which contain numerous and at the same time unusual infection techniques in the same small block of code. Its author has even managed to insert two text strings into that area, 'DRAGON-2' and 'Anti'. The first string is placed in the virus device header in the Device Name fields and, as such, is the name of that 'device'.

Dragon

Aliases:	None known.
Type:	Memory-resident and parasitic file infector with stealth capabilities.
Infection:	Any files with the EXE identifier 'MZ' which are filled with zeros from offset 0070h-0200h.
Self-recognition in Files:	Compares 28h bytes of its code with the file being accessed.
Self-recognition in Memory:	None necessary, as the stealth routine prevents a second copy of the virus being executed.
Hex Pattern (in files and in memory):	8CC8 2E01 0691 000E 0606 8CC0 488E C026 8B1E 0300 83EB 1A07
Intercepts:	Installs itself into the system as a device driver, storing the address of the virus TSR code as the address of all disk device drivers.
Trigger:	None.
Removal:	Under clean system conditions, identify and replace infected files.

VIRUS ANALYSIS 3

Nostardamus: A Virus for the Future?

‘Your hard drive is being formatted,’ reads the message on the screen - receiving such news from a virus is an unpleasant shock for a PC user. The first thought to go through his mind must surely be: ‘How long it has been since I made my latest backup?’; then, after cursing the virus’ author roundly, ‘Where could I have picked that up?’.

According to a message posted in the Russian *Fidonet* echo at the beginning of February, exactly that had happened to one unfortunate person. The user concerned was more than disconcerted to realise that this particular virus claimed it was formatting his 32MB hard drive to a new size 40MB - a ‘gift’ of 8MB from ‘Nostardamus’ (sic).

A large number of infections have been recorded by end-users in many Russian towns; indeed, this may turn out to be the next real virus epidemic. The outbreak is still local (as yet there have been no reports outside Russia) - end-users in other countries will almost certainly receive updated anti-virus programs before the virus spreads that far.

Installation, Polymorphism, and More…

Nostardamus, a 2247-byte file infector, is named after a text string contained in its body (see below). On execution of an infected file, control is passed to a polymorphic decryption loop, and the virus body is restored to its executable form.

Once control passes to the virus code proper, an ‘Are you there?’ call is performed (Int 21h, AH=F0h). If a copy is memory-resident, 4Bh in the CL register is returned. If not, the virus reserves an area of memory for itself by direct manipulation of the Memory Control Block chain, hooks Int 16h and Int 21h, traces Int 21h to find its original address, and returns control to the host program.

The virus’ polymorphic code is not complex: there are several main instructions which are selected at random from a fixed list, and diluted with junk code. This should cause little problems for most products.

However, polymorphism is rapidly becoming the norm for new viruses… and so in an attempt to confuse further, the author of Nostardamus has littered his creation with anti-debugger tricks.

First, the virus disables tracing (storing an IRET instruction at the memory location pointed to by the Int 01h vector). Next, hardware interrupts are disabled by writing FFh bytes into port 21h, and the virus code performs several dummy jumps to confuse static disassemblers.

The virus uses another anti-debugging trick, which I think is most effective, before nearly every interrupt call. Nostardamus does not keep the register values unencrypted in its body: these values are all calculated immediately prior to calls to the interrupt routine. For example, the code for an ‘Are you there?’ call is:

```
MOV AH,  C6H
XOR AH,  36H    ;  the  result  AH=F0h
Int 21H
```

The calculation is trivial to carry out, but even a routine as simple as this makes analysis by hand a time-consuming task: it is necessary to calculate register values each time the virus calls an interrupt.

The same method is used by the virus on manipulation of other constant data: for example, Nostardamus checks file names for the extensions COM and EXE, but does not keep these bytes in its body. They are also XORed, and restored just before being compared.

Int 21h Handler and Infection

The virus intercepts several Int 21h functions:

- file access functions Open File (AH=3Dh), Get/Set File Attributes (AH=43h), Load and Execute (AH=4Bh), Rename File (AH=56h)
- find functions FindFirst/ FindNext FCB/ASCII (AH=11h, 12h, 4Eh, 4Fh)
- the ‘Are you there?’ call (AH=F0h)

The virus marks infected files by setting the seconds field of their time and date stamp to 20. This is used by a semi-stealth routine on Find First/Find Next calls, which hides the increase in the length of infected files.

The virus hooks Int 24h during its infection routine. This prevents the standard DOS error message which occurs on accessing write-protected disks from being displayed. It then disables the Control-Break interruption, and checks the target file’s extension. Where it is a *.?YS (SYS) file, the virus aborts the infection routine - the author may be reserving this branch for future versions of Nostardamus.

If a file being accessed has the extension ?OM (COM), ?XE (EXE) or ?VL (OVL), control will pass to the infection routine; however, an additional check is carried out where COM and EXE files are concerned. The virus compares the file name with the string CO* (COMMAND), *EB (WEB), *NF (ADINF), *TI (ANTI ?) and AI* (AIDSTEST).

The virus aborts the infection routine if the file is COMMAND.COM. If the name derives from other names (which could identify Russian anti-virus programs), the virus switches on a special flag used on execution of an infected program. If such a file is infected, the virus erases its environment area on execution. As a result, these programs cannot locate the names of their host files to check them, nor detect whether the host file has been altered.

Files with the extension COM, EXE or OVL will cause the virus to execute its infection routine. Nostardamus first stores the file's attribute, clearing the read-only attribute, but if the System attribute is set, the virus does not infect (thus, IBMBIO.COM and IBMDOS.COM are not infected).

Then the virus opens the file, stores its date and time stamp, reads the file header and checks file length. If this is less than 1500 bytes, the virus does not infect. In the case of COM files, the virus will also abort the infection routine if the file is longer than 63288.

Nostardamus checks for specific 'identification-bytes' to prevent multiple infection. COM files will be not infected if they contain the value C3h in the byte at offset 4 from the beginning of the file; nor EXE files if the word 07B7h is present in the checksum field in the EXE header (the word stored at offset 12h of the header).

If the file is not infected, and matches the criteria outlined above, the virus calls its polymorphic code generation routine, encrypts itself and writes its encrypted code at the end of the file. It then overwrites the EXE file header with the corrected entry register values, and the beginning of COM files with the jump code:

```
MOV  REG,  OFFSET  VIRUS_ENTRY
PUSH  REG
RET
```

where REG is selected at random from the AX, BX, CX and DX registers.

Trigger Routines

Nostardamus contains several trigger routines, which are called only after more than nineteen generations of the virus have been executed. On reaching its twentieth generation, the virus' installation routine checks the system date. If the number of the month (where January is 1, February is 2, etc) multiplied by two equals the number of the day (e.g. 2 January, 4 February, 6 March), the virus displays the following message:

```
The  NOSTARDAMUS-Erase  (c)  v2.1  beta
Formating  disk  C:
40Mb
```

Next it obtains the address of the Int 13h BIOS, using undocumented Int 2Fh calls, and forces the Read Sector (but fortunately, not the Write Sector) instructions into a loop - this makes the drive light flash as if the drive were really being formatted. According to instructions in the virus code, the loop should terminate and the installation routine continue when any key is pressed. Despite this, during experiments the process caused the system to crash.

Another trigger routine is called from the file infection block of the virus' Int 21h handler. On accessing files other than those with extensions COM, EXE, SYS or OVL, the virus checks the file attribute: if it is Hidden, the virus overwrites the first byte of the file with the first byte of its own name, and gives that file a Read-Only attribute, in addition to the Hidden attribute already present.

The virus also checks the system time counter when accessing files. If the current minutes value is less than four, the virus erases the eightieth sector of the A: drive. Should the time be later than 17:59 (i.e. 18:00 or later), Nostardamus hooks Int 1Ch and displays the following message at the top of the screen:

```
HOME  RUN  !!!
```

Yet another trigger routine is placed in the virus' Int 16h handler. The virus checks that the keys are entered and bypasses (disables) F8, Shift-F8, and Ctrl-F8 keys. When Ctrl-F10 is pressed, the virus substitutes it with the F8 key.

Nostardamus

Aliases:	None known.
Type:	Memory-resident, parasitic, polymorphic file infector.
Infection:	COM and EXE files.
Self-recognition in Files:	
	COM files - the virus compares the fourth byte with C3h.
	EXE files - file checksum field (the word at offset 12h in EXE header) is compared with 07B7h (1975 decimal).
	FindFirst/Next calls - seconds value in file time stamp equals twenty.
Self-recognition in Memory:	
	'Are you there?' call with Int 21h, AH=F0h. The memory-resident virus returns 4Bh in CL register.
Hex Pattern in Files:	
	No search pattern possible.
Hex Pattern in Memory:	
	80F4 2A80 FC17 742E 80FC 6974 2980 FC61 7424 80FC 7C74 1F80
Intercepts:	Int 21h for infection and trigger routines, Int 16h and Int 1Ch for trigger routines.
Trigger:	Displays messages, erases sectors, corrupts files and keyboard input.
Removal:	Under clean system conditions, identify and replace infected files.

PRODUCT REVIEW 1

Network Security Organiser

Jonathan Burchell

This month we take a look at the *Network Security Organiser* (*NSO*) from *Thompson Network Software*. Administering a file server can be a very trying and difficult job: it is necessary to use several different and interrelated programs, and the incoherent nature of this environment can lead to configuration errors and loopholes in security. *NSO* is designed to improve the security of a network against viral infections by helping the administrator understand the current security configuration, including specific reporting of potential security holes with auditing and software distribution functions.

By highlighting potential security risks, *NSO* hopes to prevent the chance of viral infection. There are very few infectors which can deliberately circumvent *Novell NetWare* security privileges: a properly set up server should be uninfectable by anyone other than a user with supervisor privileges operating from an infected environment.

NSO is available both as a shrink-wrapped product and as shareware directly from the *Thompson* BBS. As shareware, if you decide to use it, you are asked to pay a US$400.00 per annum fee for every server on which the product is installed. Apart from peace of mind, this buys unlimited upgrades and technical support for a year.

Product Presentation and Installation

The entire product fits onto a single 3.5-inch, 1.44MB diskette. The documentation consists of a 65-page user's manual which, though well presented, is extremely short on detail and in some places no longer reflects the current version of the software.

Installation requirements for *NSO* are *NetWare* versions 3.11 and above. *NSO* does not actually install any NLMs, therefore it does not occupy server RAM. The product must process information in the bindery - no mention is made of support of *NetWare 4.0*, although it may well be that the product will work on such systems if they have bindery emulation enabled.

The install program, which is run from a workstation whilst logged on as Supervisor, handled the job of creating the required directory structure on the chosen server volume and copying the software across without difficulties. Once installed, *NSO* is started simply by typing NSO from the install directory. A single shell provides access to all the features of the package. No support for *MS Windows* is present, but the shell represents 'state-of-the-art' in DOS character-based menuing user interfaces and responds to mouse as well as keyboard input.

The major features can be grouped under four main headings: file server risk analysis, workstation inventory and auditing, electronic distribution of anti-virus software, and network-wide reporting and logging.

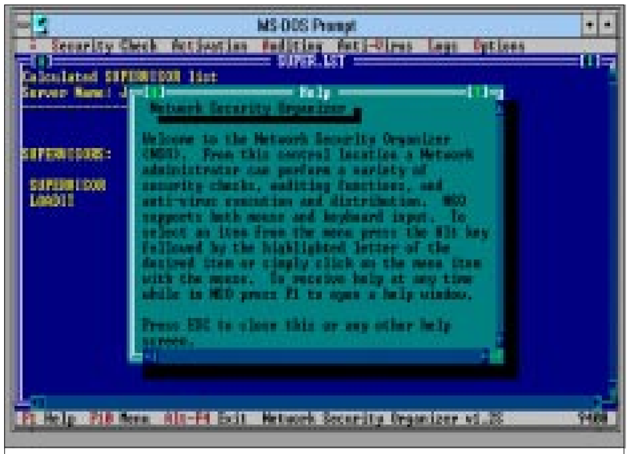
File Server Risk Analysis

The risk analysis reads the bindery and directory permissions and calculates the rights for each user and directory. The aim is to highlight every user who has Supervisor privileges, and directory/user combinations where the users have write access to a directory which contains executables - such directories are flagged as potentially infectable. Gaining this information using standard *NetWare* utilities is actually rather convoluted.

Thompson Network Software correctly points out that many networks have accounts with Supervisor privileges who have been forgotten, or which administrators may have missed (anti-virus packages for *NetWare* often create 'Superusers' for their own logging and reporting functions).

The information for the security analysis is generated by selecting the 'check server' option, allowing the choice of server to scan and volumes to investigate on the chosen server. It is also possible to specify how many levels of directory/subdirectory to probe. Usually it will be sufficient simply to check directories to one or two levels - the default case in *NetWare* is that subdirectories have the same access and privilege rights as the parent directory. Once the information has been collected, it can be displayed in a number of ways:

- **Supervisor list.** A list of all users with supervisor rights on the server. Strangely, there is no single report available from the standard *NetWare* utilities which gives this information.



```
Calculated SUPERVISOR list
Server Name: J

SUPERVISORS:

  SUPERVISOR
  LOADIT
```

```
Welcome to the Network Security Organizer
(NSO).  From this central location a Network
administrator can perform a variety of
security checks, auditing functions, and
anti-virus execution and distribution.  NSO
supports both mouse and keyboard input.  To
select an item from the menu press the Alt key
followed by the highlighted letter of the
desired item or simply click on the menu item
with the mouse.  To receive help at any time
while in NSO press F1 to open a help window.

Press ESC to close this or any other help
screen.
```

NSO provides a pro-active way of dealing with the threat of virus infection, helping close potential loopholes before they actually become a problem.

- **Security summary.** This report shows all potentially infectable directories. A directory which both contains executables and has users with write and create access is considered infectable. I would have thought the rule should be extended to include directories which have execute permission, and users with write and create access.
- **User group cross-reference report.** This shows to which group a user belongs (spelt wrongly on screen as ‘belonges’) and the members of each group. Though this has no direct bearing on security, it is a useful summary of the current user database.
- **User access analysis.** This shows the filing system by directory, listing users and groups who have access rights for each directory and detailing those rights. Not only should this be useful in improving security, but I suspect it will also be extremely useful in server maintenance. When installing a new package with several executable and data directories, it can be difficult to ensure that all users have access to the new files. The temptation is to give everyone global access - a hopeless security concept. This report should allow an administrator to tailor configuration and access rights properly.

The remaining functions of *NSO* are installed by selecting Activate from the shell. The features which may be enabled include scanning for viruses (using DOCLITE), auditing workstations, distributing anti-virus software to the workstation and grabbing log files from the workstation. Enabling a feature causes Activate to install code into the system login script which will carry out whatever has been requested at login time (Run DOCLITE, or GRABLOG, etc).

Workstation Auditing

Auditing is carried out by GATHER, a program which is able to scan the workstation and produce a report file of installed hardware and software.

The hardware report includes BIOS type, processor and coprocessor type, video adapter information, keyboard layout, the amount and type of installed RAM, installed I/O ports and the number and types of disk drives present, including their size and free space. GATHER can also detect and report on hardware and software interrupt configurations, as well as collecting a copy of the AUTOEXEC and CONFIG files.

For software, GATHER attempts to report on installed packages, by identifying a principal program name (such as WP.EXE) and a file checksum value. The list of software packages supplied is woefully small and out-of-date; however, updates to the list are available by BBS and, more importantly, it is possible to define your own additions to the database. A ‘learn’ feature helps automate this task.

The information collected from the workstation is stored in the server database, indexed by the hardware network address of the workstation and the user ID. GATHER may also be used to audit standalone PCs. The resultant log file is then copied to the server, where it can be merged with the main database. *NSO* can produce extensive reports on the audit information, which should aid in tracking the installed base of hardware and software in any organisation.

Various changes to the audit options can be made, including when to check (every login, every nth login, or randomly) and exactly what to audit. Changes in configuration (hardware or software) produce an exception report, which can be used to track problems or ‘light-fingeredness’.

“[NSO’s] ability to diagnose potential problems before they occur make it a worthwhile investment”

The auditing features of *NSO* are really rather good. With a bit of fine tuning they can be used to answer all sorts of questions, such as:

- Which users have what version of software installed?
- How much disk space is free in the company?
- If we adopt this new software package, how many workstations can currently run it and how many will need upgrading?

Software Distribution

NSO is able to distribute and keep up to date a number of workstation anti-virus packages. The current list includes *Doctor* (*Thompson Network Software’s* own product), *Dr. Solomon’s AVTK*, *F-PROT*, *IBM AV*, *McAfee Scan*, *Microsoft AV*, *Norton Anti-Virus*, *Sophos’ Sweep*, and *ThunderBYTE*. What to install is configured through the shell. The actual installation and maintenance is carried out by the NODINSTL program at login time.

For each package the shell asks what type of installation to perform (full or update), where the source files are to be found, and where to place them on the workstation.

As attractive as this software installation procedure sounds, I found it to be rather limited in practice. It seems little more than a sophisticated COPY command, and does not appear to have any product-specific data. So, whilst it will copy files over, it cannot ensure the software is going to be run - by for instance updating the users’ AUTOEXEC.BAT and CONFIG.SYS files.

Furthermore, it seems to have a very naïve idea as to which version of software is installed on the workstation. As far as I could tell, version checking is carried out by comparing a number which is stored in ‘VERSFILE’ on the user’s workstation with a number specified in the set-up program. There is no attempt made to see whether the executable (or signature file) in the source directory has changed with respect to the destination; thus any automatic update relies on the version number being edited in the set-up as well as installing the files.

NODINSTL takes its instructions from a plaintext INI file. This is the file altered by the set-up option in the shell. It can be successfully edited from DOS and it is possible to get NODINSTL to deal with completely unknown packages this way. I feel that the software distribution option may work well with *Doctor*, but is of limited use with most other products, and suspect that the concept predates the availability of true network versions of the other supported packages.

Network-wide Logging

The two sources of information for the network logs are those which come from the VBTSR and those which come from the GRABLOG program.

VBTSR is a small (about 9K of RAM) TSR which should be loaded at every workstation. It provides an element of real-time virus protection (it can check files on being opened or executed) and collects details on programs being run from floppy disk and write requests to executables. VBTSR may be set to block execution from floppy and to abort any attempt to write to an executable file.

I did not test the detection ability of VBTSR extensively, as it did not have a 'continue after detection' feature which worked; however, it seemed to miss all polymorphic infections and was not very sensitive to the other test-sets. The VBTSR signature database appeared to be in plain ASCII and had only 63 separate entries! The VBTSR could not be considered a front-line real-time anti-virus product.

The idea behind this component is that together with the associated reporting facilities in the *NSO* shell, a complete track of what external programs were run from where is maintained. Analysing such data can help pinpoint unusual activity or highlight where an infection may have originated. The information is collected from the local VBTSR file and appended to the global log file by a small utility called GRABLOG, which may also be used to grab any other log file (such as those produced by workstation anti-virus products) and integrate them into a single file on the file server, providing centralised tracking of incidents.

GRABLOG prepends each collected line with user ID and date plus an optional command-line supplied piece of text (such as the name of the utility which produced the logfile).

NSO itself does not provide extensive database reporting; however, all data files are plain text or paradox format, so it should not be difficult to construct further reporting systems.

Conclusions

As can be seen from the test results, DOCLITE is good at detection. The polymorphic detector scored 100%, apart from the Uruguay and Diet-compressed Cruncher samples: these it missed completely, giving an overall detection rate for that test-set of 83.3%. The scores on the Standard and the In the Wild test-sets are credible (96.5% and 95.4% respectively), but there is no excuse for not getting 100% of the In the Wild samples.

I find it difficult to summarise my findings about *NSO*. The security reporting feature is excellent; simply generating and using the reports could repay the purchase price. Workstation auditing features are also good, and easier to use than many equivalent products. The software distribution appears of dubious value, although it might be useful if you want to implement a protection scheme based on workstation products rather than server-based anti-virus software.

The documentation is appalling: the developers should take this very seriously. The printed manual, such as it is, is inadequate, and the on-line help is little better. Additionally, there is no sensible READ.ME documentation. Indeed, I had to guess at a lot of the functionality by invoking programs with the /? switch, then investigating the list of options.

If you are a network consultant, carrying around a copy of *NSO* might help you impress clients. Its ability to diagnose potential problems before they occur make it a worthwhile investment, and goes a long way to making up for the problems I had with the manuals.

Technical Details

Product: *Network Security Organiser version 1.20*.

Developer: *Thompson Network Software*, 2619 Sandy Plaines Road, Marietta, Georgia 30066, USA. Tel. +1 404 971 8900, Fax +1 404 971 8828.

Price: US$400.00 per server per annum, including BBS access, unlimited upgrades (also via BBS), and technical support.

Hardware used: Client machine - 33 MHz 486, 200 Mbyte IDE drive, 16 Mbytes RAM. File server - 33 MHz 486, EISA bus, 32-bit caching disk controller, *NetWare 3.11*, 16 Mbytes RAM.

Each test-set contains genuine infections (in both COM and EXE format where appropriate) of the following viruses:

Standard Test-Set: As printed in *VB*, January 1994, p.19 (file infectors only).

In the Wild Test-Set: 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, Captain_Trips, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T2, Coffeeshop, Dark_Avenger.1800.A, Dark_Avenger.2100.DI.A, Dark_Avenger.Father, Datalock.920.A, Dir-II.A, DOSHunter, Eddie-2.A, Fax_Free.Topo, Fichv.2.1, Flip.2153.E, Green_Caterpillar.1575.A, Halloechen.A, Helloween.1376, Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard, Jerusalem.Anticad, Jerusalem.PcVrsDs, Jerusalem.Zerotime.Australian.A, Keypress.1232.A, Liberty.2857.D, Maltese_Amoeba, Necros, No_Frills.843, No_Frills.Dudley, Nomenklatura, Nothing, Nov_17th.855.A, Npox.963.A, Old_Yankee.1, Old_Yankee.2, Pitch, Piter.A, Power_Pump.1, Revenge, Screaming_Fist.II.696, Satanbug, SBC, Sibel_Sheep, Spanish_Telecom, Spanz, Starship, SVC.3103.A, Syslock.Macho, Tequila, Todor, Tremor (5), Vacsina.Penza.700, Vacsina.TP.5.A, Vienna.627.A, Vienna.648.A, Vienna.W-13.534.A, Vienna.W-13.507.B, Virdem.1336.English, Warrior, Whale, XPEH.4928.

Polymorphic Test-Set: 600 genuine samples of: Coffeeshop (250), Groove (250), Cruncher (25), Uruguay.4 (75).

PRODUCT REVIEW 2

McAfee's VirusScan

Dr Keith Jackson

VirusScan, the software package from *McAfee Associates*, first appeared on the anti-virus scene in 1989, and has since held its own in a constantly changing, constantly more demanding market. The product has been reviewed in *VB* at various stages in its development (April 1991, March 1993).

Its latest incarnation heralds a major revision: this review looks at a new version of an old favourite, and asks whether it has managed to 'keep up with the pack'.

On Offer

The software came on only one type of floppy - others are available on request. Several individual components make up the product: the centrepiece is Scan, a program claiming to detect known/unknown viruses, and to remove viruses from infected files (older versions had separate programs).

VirusScan is command-line driven, but includes three graphical user interfaces which use Scan to carry out tasks (however, see below). Also provided are a memory-resident program, report manipulation features, and a *Windows*-based scheduler. *VirusScan* can be used on a network and under *OS/2*, features which were outside the scope of this review.

The 123-page A5 manual is clearly written, indexed, and easy to comprehend, but error messages are not thoroughly detailed. This is not a trivial issue, and should be dealt with. Documentation is much improved over previous versions, which were criticised by *VB* as having 'no Table of Contents, no Index, indeed very little structure at all'. This has been relegated to the past: the manual is eminently usable.

Installation

Installation involves executing the INSTALL command and specifying the subdirectory for file storage. INSTALL then copies nine files into that subdirectory, decompressing those required to operate Scan under DOS. I was intrigued to note that the memory-resident component of *VirusScan* is not inserted automatically into AUTOEXEC.BAT. Why not?

The *Windows* program installs the components which operate inside that application: a *Windows* group must be opened or created, and a new item created to execute the *Windows* version of Scan.

The manual lists three separate GUIs; for DOS, *Windows*, and *OS/2*. This review discusses only that for *Windows*, as the install program did not install the other two. In the case of the *OS/2* interface, this was unsurprising, but it was perplexing that the DOS GUI appeared not to be on the disk.

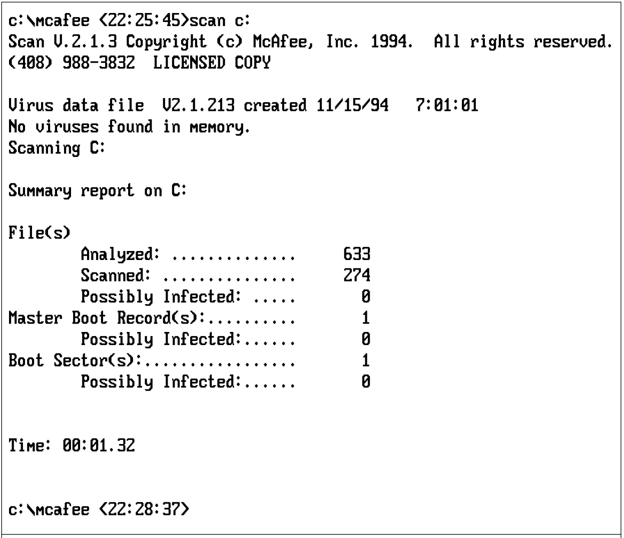
Scanning

Scanning may be operated from command-line switches or as selections made via the GUI. Customisation is available from a choice of named disk drives, executables only, within named subdirectories, and inside compressed executables. Scanning may also take place in Turbo mode, which works by 'examining a smaller portion of each file'.

So many options are provided for scanning that it is difficult to decide what to quote for timing. Using default settings, Scan took 1 minute 32 seconds to inspect the hard disk of my test PC. This rose to 2 minutes 3 seconds when all files were scanned, and fell to 44 seconds in Turbo mode. Under *Windows*, scanning took 2 minutes 15 seconds. In comparison, *Dr. Solomon's Anti-Virus Toolkit* took 1 minute 14 seconds to scan the same hard disk; *Sophos' Sweep* took 2 minutes 10 seconds in 'Quick' mode and 7 minutes 3 seconds to carry out a 'Full' scan.

A major caveat must be introduced at this point. Scanning times were reported onscreen by Scan itself: actual time taken (as measured by a stopwatch) was always larger. Scan time under default settings was 2 minutes 19 seconds (a 51% increase over onscreen time); the Turbo scan took 1 minute 30 seconds (82%). As surrounding elements are removed from the core scan time, onscreen times converge with the measured time: the figures matched most closely (a discrepancy of 7%) when the *Windows* GUI was used.

The only time which is important to the user is overall time, not a figure which omits the time taken by constituent parts. However, even allowing for the discrepancy with the time reported onscreen, Scan is still reasonably fast.



```
c:\mcafee (22:25:45)scan c:
Scan V.2.1.3 Copyright (c) McAfee, Inc. 1994.  All rights reserved.
(408) 988-3832  LICENSED COPY

Virus data file  V2.1.213 created 11/15/94   7:01:01
No viruses found in memory.
Scanning C:

Summary report on C:

File(s)
        Analyzed: .............      633
        Scanned: ..............      274
        Possibly Infected: .....       0
Master Boot Record(s):..........       1
        Possibly Infected:......       0
Boot Sector(s):.................       1
        Possibly Infected:......       0


Time: 00:01.32


c:\mcafee (22:28:37)
```

Those who are fans of DOS command line-controlled scanners will love the new version of *McAfee VirusScan*. However, those who are fans of the GUI have not been left out…

Accuracy

Scan claims knowledge of 4480 viruses. When pitted against the test-set listed in the Technical Details section, it detected 229 of the 248 virus-infected test samples, a detection rate of 92%. Curiously, this new version fails to find seven test samples successfully detected by the version reviewed in March 1993. The detection rate is noticeably poorer amongst newer additions to the test-set. Scan, in common with many other similar packages, seems to be having difficulties coping with the flood of new viruses.

The decline in detection capability also applies to detection of Mutation Engine (MtE) test samples. Previously, Scan detected all these: now it only finds 54%. This really is unacceptable - the Mutation Engine has been around since 1992: why does Scan still fail to detect it reliably?

Scan can create a report file detailing what it detects during scanning, including details of viruses and errors detected. The report file generated while scanning through the test viruses always stated in its summary that one non-critical error was detected; however, neither the scanner nor the report file indicated what the error was. I'm puzzled by this.

Validation

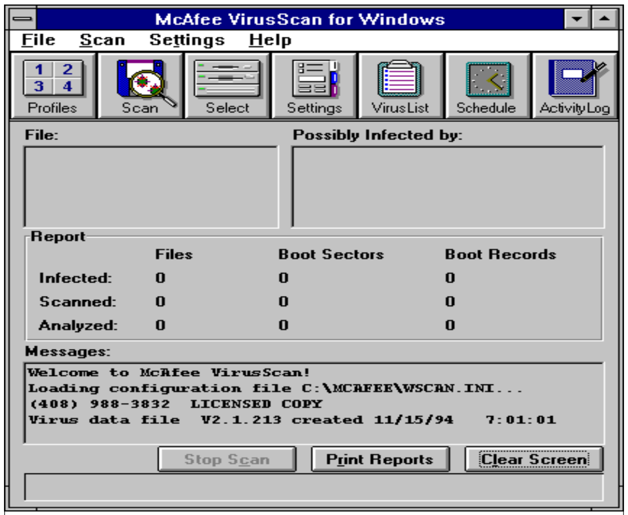
Scan 'validates' a file (checks that it has not been altered) in one of two ways: 'immunizing' by adding extra information to the file, or building a database of information about files which can be verified by future checks. This is more usually referred to as checksum verification or integrity checking.

Immunization is normally best avoided: executable files should be left in their original condition. Allowing users to introduce deliberate alterations can lead to serious problems. *VirusScan's* executable files are supplied with immunization information present, but when a software developer adds such information to his own executable files, and the process is found to cause a problem, the developer is able to sort it out forthwith. This is the only time when the recommendation to steer clear of immunization does not apply.

If users add immunization information across an entire hard disk, problems could be permanent. *VirusScan's* developers obviously recognise that immunization can in some cases cause problems, as, rightly, the creation of a database of validation information is described as the 'preferred' option. Immunization is recommended mainly for companies which distribute software to their users. Fair enough.

Creating a database of validation information while Scan was validating the hard disk of my test PC took 8 minutes 40 seconds. Subsequent verification that the files remained unchanged took 8 minutes 30 seconds.

The validation times reported onscreen (7 minutes 56 seconds and 7 minutes 46 seconds respectively) were much less than the measured times. The manual does warn that this process can take some time, and states that it may add '300% more time to scanning'. This figure is not too far out.



Report

	Files	Boot Sectors	Boot Records
Infected:	0	0	0
Scanned:	0	0	0
Analyzed:	0	0	0

Messages:

```
Welcome to McAfee VirusScan!
Loading configuration file C:\MCAFEE\WSCAN.INI...
(408) 988-3832  LICENSED COPY
Virus data file  V2.1.213 created 11/15/94   7:01:01
```

…because it comes complete with a pretty *Windows* interface, which allows easy access to the product's features.

Memory-resident Component

VirusScan's documentation states that its memory-resident component (VShield) requires 67 KB of free memory; however, it can be loaded into extended, expanded or upper memory. A memory-swapping option is available which will reduce the memory requirement to that needed by a small kernel. When expanded memory was used, only 8.5 KB of ordinary RAM was required. A multitude of options are available, so VShield's operation can be tailored to choice.

I detected a few odd quirks while using VShield. Firstly, the manual states that the option allowing VShield to monitor any access to a file works 'regardless of the file's extension'. This is not true. I had to rename the test samples to use COM and EXE extensions before the memory-resident software would detect them as virus-infected. Whenever such a file was detected, the box that popped up in the middle of the screen always overwrote the first character of the file's name, a 'feature' which does not helps users at all.

When used in its default mode, VShield detected all bar 22 of the 148 non boot-sector viruses in the test-set, a success rate of 85%. Many viruses not detected were polymorphic: when the option to detect polymorphic viruses was activated, the number detected did not increase. Also, for some unknown reason, VShield resolutely refused to install if the option to monitor a file on any access and the option to detect polymorphic viruses were activated simultaneously.

In its default state, VShield took 1 minute 16 seconds to load, a time that makes a PC reboot a less-than-scintillating experience. The operational overhead introduced by VShield was tested by measuring the time taken to start up *Norton Commander* (an *MS-DOS* '*Navigator*' program), and the time taken to copy 40 files (1.25 MB) from one subdirectory to another. Without VShield, these two actions took 0.5 seconds and 20.5 seconds respectively. When only CRC checking was invoked, the times rose to 3.8 seconds and 20.8 seconds respectively. When the file validation option was activated, the time needed to start *Norton Commander* rose to 17.2 seconds, but the file copying test still only took 21.5 seconds.

Since a similar increase in program load time applies to all executables, the PC became sluggish. Things worsened when the option to verify files on access was used: the file copying time rose to an unbelievable 9 minutes 17 seconds. I would contend that such an option is unusable. The manual claims that 'VShield adds a small amount of time to program loads and reboots', and that 'Once programs have been loaded, VShield does not degrade the performance of your system'. Given the measured times quoted above, it is being somewhat economical with the truth.

Virus Removal

Scan also has a facility to restore a file and/or the boot sector after virus-induced damage; its developers call it 'cleaning'. The product offered to clean 14% of the viruses contained in the test-sets listed in the Technical Details section. That in itself is a poor result, but of that meagre amount, how many will have been 'cleaned' correctly? All in all, not a course of action worth considering under most circumstances.

If 'cleaning' fails, the infected files may either be deleted or moved to a named subdirectory. The documentation glosses over how Scan can determine whether or not the process has been successful, although it does acknowledge that damage reversal is not possible in every case.

The safest option is *always* replacement of an infected file by a copy of the original, and I believe this tactic should, without exception, be followed as a first course of action. Therefore I will not say too much about the product's effectiveness at restoring uninfected originals.

Conclusions

Scan is simple to use, a remark which is meant as a compliment. I grow weary of reviewing anti-virus software which can rival *Windows*-based word processors and databases for complexity and the sheer number of available features. This product concentrates on performing simple tasks in a swift and useful manner. Beyond the almost mandatory graphical user interface, it forgoes frills and is better for it.

The memory-resident features offered by *VirusScan* are good at detecting viruses; far better than many other competing packages. However, this is at the expense of imposing an overhead on execution time which under certain circumstances is so large as to make it unusable.

In both previous *VB* reviews, I commented that the virus detection capabilities of *VirusScan* were excellent. Indeed, last time it was only one away from a perfect 100%. This is no longer the case. For some odd reason, Scan now fails to detect a few viruses which previous versions did, is not as good at detecting MtE viruses as it used to be, and its detection capability declines where recently discovered viruses are concerned. The first two are unforgivable. The latter is excusable, and likely to be due to the sheer numbers of viruses which are currently flooding in. Even though Scan has fallen somewhat from its previous high pedestal, it is still eminently reasonable at detecting viruses. There are much worse scanners around.

Technical Details

Product: *VirusScan version 2.1.3L.*

Developer: *McAfee Associates*, 2710 Walsh Avenue, Santa Clara CA 95051, USA. Tel.: +1 408 988 3832, BBS: + 1 408 988 4004.

UK Vendor: *IPE Corporation*, 9 Alfred Place, London WC1E 7EB. Tel.: +44 171 436 2244, fax: +44 171 916 1004.

Availability: Any *IBM-PC* or compatible which uses DOS (v3.1 or later, needs 340 KB RAM for command-line programs and 530 KB RAM for the graphical programs), *Windows* (v3.1 or later, needs 4 MB RAM) or *OS/2* (v2.1 or later, 8 MB RAM).

Price: £117.00 for a single-user licence, including unlimited updates via BBS. Price for multi-user licence on application.

Hardware used: A *Toshiba 3100SX* laptop PC (16MHz 386) with one 3.5-inch (1.4 Mbyte) floppy disk drive, 5 MB RAM, and a 40 MB hard disk, running under *MS-DOS v5.00*.

Viruses used for testing purposes: This suite of 158 unique viruses (according to the virus-naming convention employed by *VB*), spread across 247 individual virus samples, is the current standard test set. A specific test is also made against 500 viruses generated by the Mutation Engine (which are particularly difficult to detect with certainty).

The test-set contains 9 boot sector viruses (Brain, Form, Italian, Michelangelo, Monkey, New_Zealand_2, Quox, Spanish_Telecom, V-Sign), and 239 samples of 150 parasitic viruses (Spanish_Telecom appears in both lists). There is more than one example of many of the viruses, ranging up to 12 variants of the Tiny virus. The parasitic viruses used for testing are listed below. Where more than one variant is available, the number of examples of each virus is shown in brackets. For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *VB*:

1049, 1260, 12 TRICKS, 1575, 1600, 2100 (2), 2144 (2), 405, 417, 492, 4K (2), 5120, 516, 600, 696, 707, 777, 800, 8888, 8 TUNES, 905, 948, AIDS, AIDS II, Alabama, Ambulance, Amoeba (2), Amstrad (2), Anthrax (2), AntiCAD (2), Anti-Pascal (5), Armagedon, Attention, Bebe, Blood, Burger (3), Butterfly, Captain_Trips (2), Cascade (2), Casper, Coffeeshop, Dark_Avenger, Darth_Vader (3), Datacrime, Datacrime_II (2), Datalock (2), December_24th, Destructor, Diamond (2), Dir, Diskjeb, DOSHunter, Dot_Killer, Durban, Eddie, Eddie_2, Fellowship, Fish_1100, Fish_6 (2), Flash, Flip (2), Fu_Manchu (2), Halley, Hallochen, Helloween (2), Hide_Nowt, Hymn (2), Icelandic (3), Internal, Invisible_Man (2), Itavir, Jerusalem (2), Jocker, Jo-Jo, July_13th, Kamikaze, Kemerovo, Kennedy, Keypress (2), Lehigh, Liberty (5), LoveChild, Lozinsky, Macho (2), Maltese_Amoeba, MIX1 (2), MLTI, Monxla, Murphy (2), Necropolis, Nina, Nomenklatura (2), Nuke_Hard, Number_of_the_Beast (5), Oropax, Parity, PcVrsDs (2), Perfume, Pitch, Piter, Polish_217, Power_Pump, Pretoria, Prudents, Rat, Satan_Bug (2), Shake, Sibel_Sheep (2), Slow, Spanish_Telecom (2), Spanz, Starship (2), Subliminal, Sunday (2), Suomi, Suriv_1.01, Suriv 2.01, SVC (2), Sverdlov (2), Svir, Sylvia, Syslock, Taiwan (2), Tequila, Terror, Tiny (12), Todor, Traceback (2), Tremor, TUQ, Turbo_488, Typo, V2P6, Vacsina (8), Vcomm (2), VFSI, Victor, Vienna (8), Violator, Virdem, Virus-101 (2), Virus-90, Voronezh (2), VP, V-Sign, V-1, W13 (2), Willow, WinVirus_14, Whale, Yankee (7), Zero_Bug.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel. 01235 555139, International Tel. +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email virusbtn@vax.ox.ac.uk
CompuServe 100070,1340@compuserve.com

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel. +1 203 431 8720, Fax +1 203 431 8165

END NOTES AND NEWS

Precise Publishing Ltd has announced that it is to join the ranks of anti-virus software vendors/developers staging virus workshops. Their first ***Live Computer Virus Hands On Workshop*** is on 26 April. The one-day event costs £395, and attendees will take away the anti-virus software used on the course. Further information is available from the company on Tel. +44 (0)1384 560527, fax +44 (0)1384 413689.

The *NCSA* (*National Computer Security Association*) has announced the appointment of Dr Peter Tippett (formerly of *Symantec*) as President and Executive Director. Other recent departures from *Symantec* include virus specialist Joe Wells (to *IBM*) and *NAV* Product Manager Therese Padilla (to *Command Software*).

Due to high demand, *Sophos plc* is organising **an extra anti-virus workshop on 26/27 April 1995**. Day one is the *Introduction to Computer Viruses*, and Day two, the *Advanced Virus Workshop*. One single day costs £325; both together are priced at £595. Contact Karen Richardson at *Sophos* for details: Tel. +44 (0)1235 559933, fax +44 (0)1235 559935.

RG Software has announced the **appointment of former *VB* contributor Mark Hamilton** as its Marketing Director. *RG Software* develops *Vi-Spy*, a well-known and reliable US anti-virus package. Further information on the product, the company, and/or Hamilton's capacity there are available on Tel. +1 612 423 8000.

CERT has warned of a **new threat to *Internet* security** which allows an intruder to create packets with spoofed IP addresses. This exploits applications which use authentification based on IP addresses and leads to unauthorised user and possibly root access on the targeted systems. In the current system attack pattern, intruders may dynamically modify the kernel once root access is obtained. A copy of the *CERT* advisory is available via anonymous ftp from ftp.cert.org.

S&S International has announced the launch of a **new version of its *Anti-Virus Toolkit for DOS and Windows***. Some of the enhancements in version 7.0 are an even more user-friendly graphical user interface and archive and compressed file support. For further details, contact *S&S* on Tel. +44 (0)1296 318700, fax +44 (0)1296 318800.

It seems the United States is having a difficult time **fighting software piracy**: in two recent court cases, one was dismissed, and the other ended in an out-of-court settlement (although this latter effectively prevents the accused from continuing his alleged activities). A Federal Court judge has dismissed charges against David LaMacchia, who was accused of running a BBS accessible through the *Internet* from which pirated software could be downloaded. In Minneapolis, Aaron Chung agreed to pay upwards of US$29,000 in fines and to surrender computer equipment worth US$25,000 and software valued at between one and five million US dollars. Chung had been accused of running a BBS from which subscribers could download pirated software.

Sophos plc has increased the number of systems supported by its ***InterCheck* anti-virus technology**. It is now available for use with *AS400*, *Banyan Vines*, *HP-UX*, *LANtastic*, *OSF-1*, *Solaris*, and *AIX*. Contact Karen Richardson at *Sophos* for further information - Tel. +44 (0)1235 559933, fax +44 (0)1235 559935.

EuroSec 95, the sixth European forum on IT 'Quality and Security', will be held on 22/23 March 1995 at the *Hôtel Lutetia* in Paris, France. The conference will discuss **developments and needs in terms of IT security products**. Contact Isabelle Hachin: Tel. +33 (1) 4289 6566.

S&S International will be holding its next anti-virus workshop on 15/16 May in Berkhamstead, Hertfordshire, UK. The two-day session offers hands-on experience in managing and recovering from several types of virus. Tel. +44 (0)1296 318700, fax +44 (0)1296 318800.