ISSN 0956-9979 JUNE 1994

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Pathogen revisited.** Just how much of a risk is this virus, and how concerned does the user need to be? A review of the spread of the virus and the real risk posed to users is given on page 3.

• **Source code viruses.** The new generation of viruses infect source code, spreading via non-executable objects. What are the implications for vendors?

• **Heuristics.** How much of an advantage is it to use a scanner which contains heuristic elements? Are heuristics likely to take over from conventional, virus-specific detection equipment? For an explanation of the technique, and a review of some of the available products, see page 12.

CONTENTS

EDITORIAL

Viruses for Sale, a Dime a Dozen

The latest move by virus supremo Mark Ludwig (of *Little Black Book* infamy) is set to make many people sit up and take notice: *American Eagle* has published a CD-ROM packed with fully working samples of computer viruses, virus creation toolkits, and malicious software. Although viruses have been available in limited quantities for some time, this is the first large collection made publicly available, and has some interesting implications for the anti-virus industry.

The CD-ROM is priced at $100 and, should it sell in any quantity, will make Ludwig a tidy profit (once compiled, each additional CD costs peanuts to produce). However, how many people will be prepared to pay for the collection? Unfortunately, finding prospective buyers will not be too difficult. One hundred dollars is not a vast sum, and there are doubtless plenty of rich schoolchildren who will be drawn to the 'outlaw' image portrayed by Ludwig.

" it is highly ironic that those who are lining Ludwig's pockets with gold are his largest and most powerful opponents "

Inquisitive minors are not Ludwig's only potential customers, nor his most lucrative. This latest offering has two guaranteed buyers: the anti-virus vendors and the large-scale users of anti-virus software. The vendors will justify their purchase on the grounds that they must protect the users. Having spoken to a number of anti-virus researchers on the subject, the overall conclusion is that many feel obliged to buy the CD in order to ensure that their customer is protected. To quote one well-known scanner manufacturer: 'I feel ethically forced to spend money on this thing.' For users, the argument is just as pervasive: this rather seedy offering is out there, and they want to know what is on it, in order to evaluate the threat posed to their organisation (not to mention the chance to test their chosen product against some live viruses).

It is not clear how sound either of these arguments is, but it is highly ironic that those who are lining Ludwig's pockets with gold are his largest and most powerful opponents. The ethical questions are complex, but one thing is for certain: the likely reaction of the industry ensures the market for the next edition of the collection.

Other than a mad dash of virus collectors rushing out to buy the CD, it is worth reflecting on the consequences of Ludwig's actions. Firstly, the CD could become the *de facto* test-set used by PC magazines - if the industry cannot supply an unbiased test-set, maybe Ludwig can. Come to think of it, this may be the answer to the UK government's *ITSEC* problems (see *Virus Bulletin*, July 1993, p.2). Moreover, even if a sample on the CD is *not* a virus, an unscrupulous firm could report that it is infected anyway - the user has no way of interpreting the test results, so it will simply be a case of the scanner with the highest score winning.

Secondly, anyone with a modem could put together a rudimentary collection of viruses. If one knows where to look, samples are easily obtained, yet there are still only a small number of variants in the wild. The main risk of the CD is that it spreads virus code around a wider audience than ever before - the most probable result of which will be the unintentional infection of some of its buyers.

Meanwhile, the industry pays its thirty pieces of silver, in the name of protecting 'the good of the many'. This is uncomfortably close to paying the virus authors for their handiwork - the equivalent of a glazier giving a 16 year old a brick, muttering about how breakable windows look, and telling him there will be a £20 commission on each customer referred by him. The question of where to draw the line between keeping up with industry developments and actually encouraging virus writing is a thorny one, but wherever the line lies, the purchase of this CD is perilously close to it.

As the trend for wider dissemination of virus code continues, it is rapidly approaching a time when anyone who wants a virus can get one. If things degenerate further, maybe the least painful route would be for the industry to offer users the viruses which it wants them to have: 'Come on Sir, roll up for the Virus Service. Simply send your $99.99 and receive the virus test-set of your choice. Better yet, guv'nor, for the discerning customer, why not buy the entire collection. Yours for only a dime a dozen… and I'm cutting my own throat.' But then again, aren't we all?

NEWS

Pathogen: Storm in a Teacup?

Most UK readers will already be well aware of the storm of publicity surrounding the SMEG (aka Pathogen and Queeg) viruses. However, despite making national news, it is important to evaluate the true extent of the threat.

The number of reports of Pathogen varied from vendor to vendor, with the largest number coming from *S&S International*, who issued a press release warning users of 'a considerable threat to unprotected computer users'. The *S&S International* grand total (claimed to be 'about 12'), is larger than the number of reports to all other sources combined, with *New Scotland Yard* and *Virus Bulletin* having two reports each, and *Sophos* one. Given that many of these statistics are already known to overlap, a reasonable estimate of the number of sites affected is at most about 15, making the 'Security alert' press releases put out by *S&S International* and *Secure Computing* wholly unnecessary.

Despite different opinions about how widespread the problem is, Pathogen is certainly 'in the wild'. A straw poll of the larger vendors showed that a number had issued updates for their products, including *McAfee Associates, S&S International, Sophos, Frisk Software International* and *ESaSS*. Those currently unable to supply a detection algorithm included *Central Point*, *Intel*, and *Symantec*.

The other company contacted, *IBM*, informed us that their new version (1.06), due out at the beginning of June, will also be able to detect the virus. According to Sue Ling from *IBM*, the company felt no need for undue panic, and did not view Pathogen as an issue to cause serious concern. This feeling was reflected by a number of other vendors: neither *Central Point*, *Symantec*, *IBM* or the *NCSA* received any calls from customers who experienced the virus first-hand.

By far the most amusing aspect of the entire media circus was a report in *PC Week* claiming that the company *Gateway 2000* had shipped 70 machines which contained software infected with a virus which was 'not the so-called Smeg polymorphic virus which has troubled other manufacturers…' Further investigation showed that the *PC Week* article had arisen due to a game of Chinese whispers played between *Gateway* and its PR company, *Text 100*, which seemed to have confused the word 'bug' with the word 'virus'. Surely this is an exception to the rule of 'no publicity is bad publicity'. No machines were, in fact, infected.

It is true that where a virus is found 'in the wild', there *is* cause for concern: however, this concern must not degenerate into free-for-all panic. It is extremely rare for a new virus to leap to the top of the virus prevalence table, and new viruses are discovered in the wild all the time. Pathogen is no exception: although it presents above-average detection problems, the entire incident seems to have been no more than an industry-provoked storm in a teacup ∎

Virus Prevalence Table - April 1994

Virus	Incidents	(%) Reports
Form	15	41.7%
JackRipper	3	8.3%
New_Zealand_2	3	8.3%
Exebug.4	2	5.6%
Form.b	2	5.6%
Spanish_Telecom	2	5.6%
Cascade	1	2.8%
EXEBug.1	1	2.8%
Form.II	1	2.8%
Liberty	1	2.8%
Macgyver.2083.b	1	2.8%
Parity_Boot.A	1	2.8%
Penza	1	2.8%
V-Sign	1	2.8%
Viresc	1	2.8%
Total	36	100.0%

NCSA on CompuServe

The *National Computer Security Association* (*NCSA*) has announced the establishment of its own forum on *CompuServe*, which is dedicated to coverage of information security and computer ethics.

One section, the InfoSecurity News, will provide up-to-date information about security incidents, and is marketed as a 'valuable resource for journalists covering the field'. Other individual sections which are available include topics such as PC and LAN Security, UNIX/Internet Security, Disaster Recovery, Mainframe Security, Telecom Security, Encryption, and Anti-Virus Support.

The anti-virus support forum will be co-moderated by *Virus Bulletin*, and aims to provide a vendor-independent forum for the discussion of anti-virus software and all other matters pertaining to viruses. Any vendor or reader is invited to post messages in this sub-forum.

Informational resources available through the *NCSA* include books, research reports, training materials, conference proceedings, and tools, all of which are featured in an information security resource catalogue. This is available from the organisation free on request.

The *National Computer Security Association* forum can be accessed from any *CompuServe* command prompt by entering 'GO NCSA'. Private Email to the *NCSA* should be sent to 75300,2557@compuserve.com, and to *Virus Bulletin* on 100070,1340@compuserve.com ∎

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 20 May 1994. 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
- **D** Infects DOS Boot Sector (logical sector 0 on disk)
- **E** Infects EXE files
- **L** Link virus
- **M** Infects Master Boot Sector (Track 0, Head 0, Sector 1)
- **N** Not memory-resident
- **P** Companion virus
- **R** Memory-resident after infection

Ash.449, Ash.1586

CN: Similar to Ash.451. There is also another, seriously flawed Ash variant, 1586 bytes long. It appends its code to the host file, but does not put a JMP at the beginning of the file, so the code is never executed.

```
Ash.449         8DB6 0501 BF00 01B9 0400 FCF3 A4B4 1A8D 96C6 02CD 21B4 4E8D
Ash.1586        E802 00EB 208A 8637 078D B635 01B9 0006 3004 D2C0 46E2 F9C3
```

Better_World.E

ER: A minor variant detected with the Better_World (previously Fellowship) search pattern.

Budo.B

CR: A new variant of this Finnish overwriting virus. Detected with the Budo pattern.

Burger

CN: New variants of this primitive overwriting virus are 505.K, 505.L, 505.M, 505.N, 512.B, 560.AO, 560.AP, 560.AQ and 560.AR. All are detected with the Burger pattern.

Cascade

CR: Three new variants have now been reported (1701.R, 1704.T and 1704.U), all of which are detected with the Cascade (1) pattern.

Creeper.472

CR: Very similar to the 476-byte variant.

```
Creeper.472     0E0E 071F C3CD 2050 2D00 4B74 2658 3D00 0775 15A1 8A01 8BF0
```

Curse_IV

CER: A Dutch 400-byte virus belonging to the small group of those found 'in the wild'. It contains the text 'CURSE IV - Dedicated to Eve'.

```
Curse_IV        80FC 4B74 03E9 0601 501E 5206 53B8 023D CD21 50B8 0057 5BCD
```

Dark_Avenger.1800.Satan

CER: Contains the text 'Written by Mad Satan in TAIWAN'. Detected with the Dark_Avenger pattern.

Diamond.1050

CER: This variant seems to be based on one of the original 1024-byte variants, but is slightly longer. An unremarkable variant, detected with the Diamond search string.

Doom_II.1249

CER: A 1249/1261-byte variant, detected with the Doom2 pattern.

Ear.Ear.B

C(E)R: This variant is very similar to the virus that was originally reported as Ear-6, and is detected with the same pattern. It contains a flaw in the code, however, so will not infect EXE files correctly, causing file corruption. There is also a new .C variant, which works correctly.

Frodo.Fish_6.E

CER: A minor variant. The decryption loop has been slightly altered to invalidate earlier search strings.

```
Frodo.Fish_6.E  E800 005B 81EB A80D B958 0D2E 8037 ??43 E2F9 2EFE 8FB3 0074
```

Fumble.867.F

CR: Another minor variant, detected with the Fumble (previously Typo-COM) pattern.

Genesis

CR: A 504-byte virus, distributed in source code form. The author calls himself 'Holy Spirit', and a comment in the source code gives the name of the virus as 'Genesis 1.0'.

```
Genesis         FEC4 3D00 4C74 03E9 B700 5053 5152 5657 061E FA33 C08E C026
```

HLL.7940

CN: A non-destructive HLL virus which adds 7924 bytes to the beginning of programs it infects, and 16 bytes to the end. No search string will be given, because of the high risk of false positives. Other parasitic HLL viruses this month are HLL.3677 and HLL.3678.

HLLC

P: Four new Pascal or C 'companion' viruses have been discovered: HLLC.Christmas (6888), HLLC.Even_Beeper.D, HLLC.Globe.7705 and HLLC.Sauna (8224).

HLLO.3816

CN: This family is somewhat artificial, as the viruses belonging to it have nothing in common other than being written in Pascal or C and overwriting files they infect. As overwriting viruses, they have virtually no chance of spreading. This particular 3816-byte virus is written in Turbo C. Other HLLO viruses not previously mentioned by *VB* are HLLO.3800, HLLO.GOV (EN) and HLLO.Shadowgard (CEN).

Jerusalem.1808.Executing **CER:** The only unusual thing about this variant is that it contains the text 'Executing COM files...'. Detected with the Jerusalem-US pattern.

Jerusalem.1808.Frere.I **CER:** Detected with the Jerusalem-1 pattern. This pattern will also detect a new 1506/1511-byte variant.

Jerusalem.AntiCad.2454 **CER:** This might be the oldest variant of the AntiCad group, although discovered only recently. It is detected with the ACAD-2576 pattern, which also detects a 2656 variant containing the text 'G Dengue'.

Jerusalem.Pipi.1536 **ER:** Contains the text 'PI-PI', and is detected with the Pipi search string, as is the 1552-byte variant reported in October 1992.

Jerusalem.PSQR.Satan **CER:** By the author of the Dark_Avenger.Satan virus, and containing a similar text string. Detected with the PSQR pattern.

Jerusalem.Smile **CER:** This 2576/2587 byte-variant from Taiwan contains the text 'Smile Again'. Detected with the Jerusalem-US pattern.

Jerusalem.Sunday.Nai-Tai **CER:** A Taiwanese variant, which does not infect EXE files correctly. Detected with the Jerusalem-1735 pattern. The same pattern will also detect the new Sunday_II.B variant.

Jihuu.686 **CN:** A Finnish virus. Very similar to the 621-byte original variant and detected by the same search string.

Leprosy.Sandra **EN:** Yet another variant of this overwriting family, 682 bytes long.

```
Leprosy.Sandra  BA00 01CD 21E8 0100 C3BB 4101 8A27 3226 0601 8827 4381 FB58
```

Lockjaw.Flagyll.371 **ER:** An overwriting, 371-byte virus.

```
Flagyll.371     9C06 1E50 5352 3D00 4B75 03E8 0B00 5A5B 581F 079D 2EFF 2E73
```

Old_Yankee.1961.B **EN:** This variant is detected with the Old_Yankee pattern, and is similar to the original variant. However, despite the fact that it is the same size, it cannot be disinfected in the same way. There is another variant, 1961.C, which is flawed, as it seems only to infect the first file it finds, but will do so many times.

Pixel **CN:** Several unremarkable variants have been discovered recently, all of which are detected with the Pixel-936 pattern. They are 739, 846.B, 851 and 1268.

Proto-T.694 **CR:** Detected with the Proto-T pattern. Three other recent Proto-T variants require separate search patterns. The Proto-T virus family is awaiting better analysis and classification, as it is possible it should be merged with another Dutch family of viruses.

```
Proto-T.1050    1E06 5756 5053 5152 3D00 4B75 0D2E 8C1E 2905 2E89 162B 05EB
Proto-T.1053    1E06 5756 5053 5152 3D00 4B75 0D2E 8C1E 2C05 2E89 162E 05EB
Ritzen.1087     5053 5152 1E06 5756 9380 FF3D 7414 81FB 004B 740E 5E5F 071F
```

PS-MPC **CN, CEN, EN:** This month brings the following variants: G2.573.C (CEN), Pikninny (CEN, 616), Powermen (EN, 717) and Small_ARCV.B (CN, 236).

Skew **CR, ER:** Both viruses in this family are very similar, with one important difference: the 445-byte variant only infects EXE files, but the 458 one will only infect COM files.

```
Skew.445        0657 1E50 5152 5653 558B EC33 C98E C180 FC4B 7413 FA26 C706
Skew.458        0657 1E50 5152 5653 558B EC80 FC3D 7522 8BFA B93F 0047 803D
```

SVC **CN:** There are several new variants of this East European virus. Two are detected with the SVC_5.0 pattern. Three new variants, 2936, 3112 and 3241 bytes long, require new patterns.

```
SVC.2936        5153 502E A327 0B2E 813E 270B 004B 741B 80FC 3D74 1980 FC3E
SVC.3112        5606 86E0 35FF FF8E C00E 1F33 FFB9 A20B FCF3 A607 5E74 03E9
SVC.3241        5153 502E A358 0C2E 813E 580C 004B 741B 80FC 3D74 1980 FC3E
```

Sybille **ER:** Two variants of this family are known, 858 and 1200 bytes long.

```
Sybille.858     3D00 4B75 F350 5351 5256 5755 1E06 1E52 0E1F E8A9 015A 1FB8
Sybille.1200    7503 E990 03B8 00F0 8B16 7604 CD2F 4174 CC0A C075 C8B8 0158
```

VCL **CN:** Several new encrypted VCL-generated viruses: Dial.671, Diarrhoea.1221, Pro-Choice (1569) and Reptoid (2536). Most are of little interest, excepting perhaps the Dial.671 variant, which attempts to dial the number 1-900-976-6274. There are also some unencrypted variants this month: 514, 534, 660, 2750, 3243 and Mimic.4863. Finally, there are a few overwriting variants: 356, 418, 509, 541, Cockroach (614 bytes) and Jam (458 bytes), as well as two companion viruses: 604 and Heevahava.516.

VCS.Standard.Bad_Poem **CN:** An unremarkable variant, containing an extremely bad poem. Detected with the VCS_1.0 string.

Vienna.608.B **CN:** Detected with the Vienna-4 pattern. This pattern will also detect a new 526-byte variant, which does not work properly, as it does not store the original first three bytes of infected files anywhere.

Virdem.1336.Killer.C **CN:** A minor variant, detected with the Virdem pattern.

INSIGHT

Bob Bales - The NCSA

The *NCSA* (*National Computer Security Association*) was founded in 1989, to tap a market in the area of PC and Local Area Network security. One name has become synonymous with the organisation: Bob Bales, who has been involved with the *NCSA* since 1990, shortly after the group's inception by David Stang.

When he met Stang, Bales was running a consulting group specialising in systems integration and software development. The professional relationship which evolved led to Bales' group doing work for Stang, and vice versa. Bales' move to the *NCSA* happened soon after.

The Early Days

Part of the *NCSA's* challenge was to distinguish between what they and other security organisations of the time were doing. The major difference was that many of the others had grown up with mainframes and were having difficulty converting to PCs and LANs: 'We were approaching the problem from the ground up,' explained Bales, 'and they, from the top down. We met in the middle - we overlap now, but feel that the *NCSA* is still in many ways unique.'

Since then, the *NCSA* has grown apace. It has built a network of support professionals, and membership has grown from around 100 companies to over 1500 - Bales sees this as indicative of the evolution in computer security awareness being experienced by many organisations.

Aspirations and Achievements

A primary function of the *NCSA* is to act as a 'clearing house', providing information generally unavailable to network administrators. '*NCSA* membership varies from small networked offices to industry giants, and we think we fill that niche pretty well. If you plot the largest and the smallest companies on an economic line, you can see that there are about 100,000 companies in between!' Bales said. 'We see this area as our domain. Our emphasis is on the LAN manager, or the person managing computer security as a collateral duty.'

The *NCSA* offers many services: a bulletin board, a newsletter, a *CompuServe* forum, and telephone support for members with questions on computer security (in particular, viruses). Other services include seminars and conferences discounted to members, and there is a catalogue full of resource materials which are available at discount prices.

The organisation also certifies anti-virus products, based on detection rates. Present certification is pass/fail: 90% of the test-set must be found, or the product fails. Vendors have full access to the virus test samples - Bales feels this openness is justified: 'We're primarily an end-user organisation; we feel that if we construct a test encouraging vendors to do well on it, end-users will be better protected by products which participate.'

Another of the *NCSA's* concerns is the *AVPD* (*Anti-Virus Product Developers*), set up in 1991 to develop areas of common concern and interest in the industry. It was agreed that non-competitive issues would be addressed, such as development of a standard library for testing purposes, a virus naming convention, and a code of ethics for developers. For several years, this cooperation was the only formal point of contact between vendors, and contributed to the emergence of a degree of concordance amongst them.

The Virus Issue

The incorporation of anti-virus software into *MS-DOS 6* has raised awareness of the problems inherent within the system. Bales declared himself pleased with this: 'It's good, from an awareness standpoint - if Bill Gates says it's a problem, people accept it. *MS-DOS 6* now seems to play only a small part in most people's anti-virus strategy, but on balance, I think it's a good addition.

'Soon every PC will have anti-virus capabilities. In our 1991 survey we found that about 43% of infections in businesses occur as a result of disks brought from home - if we can detect infections in the home, it's good for business.'

"with a uniform set of standards, we may be creating the blueprint for uniformly-developed malicious code"

He feels strongly that prevention is the way ahead in the fight against viruses. Although the off-line scanner is still important, Bales believes that a good TSR is more effective, providing a line of defence preventing infection, where scanners look at the environment after the fact.

It is well-known that most users see virus prevention as synonymous with virus scanners - this preconception is present among *NCSA* members too. He believes that action is needed to rectify this: 'Right now, scanning gets big play, but as an industry, we need to do a better job of communicating the virtues of other approaches.'

He thinks that server-based products which integrate with the workstation product will become the main anti-virus vehicle in most companies, and views the next step as comparable products for systems other than *NetWare*.

Legal Queries

The *NCSA*, and Bob Bales in particular, often have contact with people who have their data destroyed by viruses, or have other business problems caused by them. Laws on computer viruses tend to be an unclear issue: in the USA, due to the First Amendment, people are loath to rescind the supposed freedom to which virus authors are entitled, a fact reflected in the virus-writing competition currently being promoted by Mark Ludwig. How does Bales feel when he sees such contests happening, legally and openly?

'It's outrageous. One estimate says that, since 1990, computer viruses have caused one billion dollars damage and expense. In today's economy, we can't afford that kind of business overhead, so we think that such a contest, although not illegal, is totally irresponsible. While the First Amendment guarantees people like Ludwig the right to do the things they're doing, it also protects us in decrying those sorts of activities. We intend to try and evoke a sense of moral outrage at such behaviour.'

He believes, however, that any legal jurisdiction would be difficult, intertwined as the issue is with the American right to freedom of speech. Bales believes that the best solution would be to work on legislation which deals with the aftermath, so that people who write viruses will pay. The law needs to be modified so that when damage occurs, the virus writer is responsible for the result of his actions

'It's difficult to prosecute a virus author,' Bales explained. 'You must first prove that he wrote the virus, and also that he intended the damage which resulted in a trashed hard disk. That's virtually impossible. Something like the *UK Computer Misuse Act*, which would describe virus activity as unlawful access, would be useful.'

This is only one of the many difficulties: 'I've spoken with Scott Charney, who heads the *Computer Crime Unit* at the *US Department of Justice*. He believes that modification of existing law would help with prosecutions. Of course, there is another phenomenon - the typical age of the average hacker or virus writer. If you're confronting a fourteen-year-old kid who's responsible for the incident, the justice system tends to be merciful - public floggings are out of style!'

Part of the problem, in his eyes, is that it is not possible to put a value on data, although initiatives to try and define some standards have begun.

'It's ironic,' said Bales. 'If you can prove damage - to do this, you have to reconstruct the data, and keep labour records associated with that effort - then you can collect damages. There are cases where that has happened. The better a job the network manager does, the less punitive the damages: if all you have to do is reload all the data from last night's backup tape, then there wasn't much damage. That's an imbalance in the way which that particular law is administered which just isn't equitable. An incompetent company can collect major damages, and somebody who's doing a good job is... stuck.'

Bales: 'One estimate says that, since 1990, computer viruses have caused one billion dollars damage and expense. In today's economy, we can't afford that kind of business overhead'.

Planning Ahead

The *NCSA* has become well established in the USA, and Bales has many plans to expand its presence. He aims to become more focused on the areas of training, consulting, and publishing. New projects include creation of multi-media educational products, and production of conference proceedings in multi-media.

Another indication of this wider viewpoint is its new *CompuServe* forum (accessed by typing GO NCSA within *CompuServe*), intended to be a meeting place for people concerned with computer security. Topics will include, amongst others, computer crime, law, and ethics, viruses, *UNIX* security, communication security, encryption, access control, and business resumption planning. The various sub-forums will be managed by industry 'names'.

The current US administration's emphasis on the national information infrastructure (NII), the so-called Data Superhighway, and health care reform, will soon catapult digital communication into the limelight. However, Bales believes that this development will bring more problems - the more widespread the technology, the higher the probability that it will come under attack. One such danger is in the NII: it will, he thinks, become a set of standards for information interchange. Many systems will be lashed together, and more emphasis will be put on interoperability.

'Once you've done that,' he explained, 'you've created an environment where a virus might move and thrive. Today, due to the fragmented nature of our infrastructure, things don't interoperate, so it's difficult for them to move about and propagate. With a uniform set of standards, we may be creating the blueprint for uniformly-developed malicious code, be that worms, Trojans, viruses, or what have you. So, there is a downside to the expansion of technology.' However, Bales says that, come what may, the *NCSA* will still be there to help: 'We will guarantee a first line of defence - we always have, and we always will.'

VIRUS ANALYSIS 1

Finnish Sprayer: Electronic Graffiti

Mikko Hyppönen
Data Fellows, Finland

Virus writers have sometimes been compared to people who create graffiti. It is as difficult to find a rational motive for vandalising other people's property with sloppy spray-paintings as to understand the rationale behind creating a harmful computer program. Whatever the reason, it is unfortunate that both of these activities remain popular. In the case of Finnish Sprayer, one can see a person who perhaps combines both of these pursuits - the 'artist' scrawls his electronic graffiti over the entire contents of an infected hard drive.

This virus was first found in Finland in December 1993, and has quickly spread throughout the country. It was not long before it was found in Sweden, Russia and Estonia, and it may well have spread even further.

Installation

Finnish Sprayer is a fairly typical boot sector virus which infects floppy boot sectors and hard disk Master Boot Sectors (MBS). It employs stealth methods to conceal its presence, and contains two destructive trigger routines.

The virus stores the original boot sector and its own code on the last three sectors of either the active hard disk partition or a diskette. When a PC is booted from an infected disk, the virus code is executed, and either installation or hard drive infection begins.

Its first action is to load the second sector of its code from disk. After this, it relocates all of its code to the top of the conventional memory area and continues the execution from there, decreasing the available memory by 5K. The reason for reserving this large area is unclear - the virus only requires 1K of memory to function, making it likely that this is a simple arithmetical error.

The next part of the virus code is also rather unusual: it checks whether or not the operation of moving the second part of the virus code to memory has been successful. This is done by searching for the letters 'Ai' in the area into which it believes it has loaded the code. If this marker is not found, the virus will try to overwrite the first sector of the hard disk with random data, and reboot the machine.

This destructive routine does not work because of a programming error, but it is obviously meant to be executed if a read error occurs during the virus' installation phase, or if the second part of the virus code is corrupted.

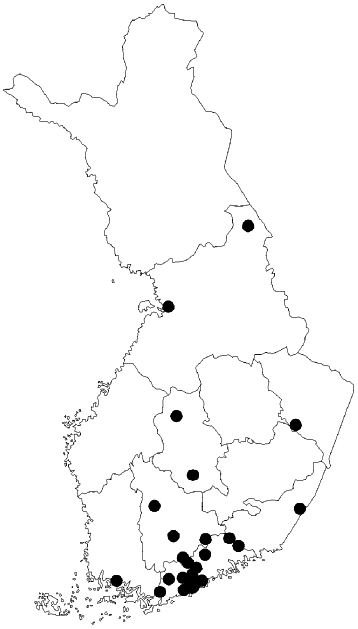
Propagation

The virus is now ready to infect the MBS of the hard disk. It loads the MBS to memory, and analyses the partition information. The virus then searches for its own infection marker: the letters 'Ai' in offset 45h of the MBS. If the computer is already infected, the virus code exits, and lets the normal boot process continue.

If the machine is uninfected, the active partition is located, and its file system type is identified - only partitions using a known DOS file system are infected. This kind of checking is rare among boot sector viruses, and means that the virus will not infect PCs running other operating systems (e.g. *OS/2* and *Windows NT*).

If the hard disk is found suitable for infection, the virus moves the original partition information to the correct place inside the viral code and writes an image of its first sector to the MBS. The virus calculates the location of the end of the active partition and copies the original MBS and the second part of its own code to the last two sectors. Any data contained within these sectors is overwritten.

The virus then executes the original boot sector. It does not stay active after the initial infection (i.e. when a previously uninfected computer is booted from an infected diskette), and will only function when the machine is booted from an infected hard disk.



The Finnish Sprayer virus has been reported throughout Finland, with most incidents concentrated in the south of the country. Reports are indicated by black dots.

Stealth

Once the hard disk is infected, the virus code will be executed during every boot-up. Its operation is similar to that of the floppy disk boot code, but after initialisation it makes a date check, and collects the original Int 13h vector for later use. Finnish Sprayer then installs its own Int 13h handler by modifying the interrupt table in low system memory to point directly to the virus code.

The virus' Int 13h handler starts by checking the called function. If this is not a disk read, the virus passes the call on to the original Int 13h handler, otherwise a rudimentary check is made to ascertain whether the call is directed to a fixed disk or a floppy drive. This information is used to branch program flow.

In the case of a read from hard disk, the virus checks whether it is a request to read the MBS. If so, the register values will be replaced with the head/cylinder/sector values of the original MBS and control is passed to the original Int 13h handler. The BIOS routines then complete the stealth operation by reading the original MBS to memory, and returning it to the calling function.

The virus does not stealth the last two sectors of hard disks, and does not stealth floppy disks at all. If the intercepted Int 13h call is a read from a floppy disk, the virus checks whether or not the disk is already infected. If it is not, the virus inspects the 'total number of sectors' field in the boot sector in order to to ascertain the diskette type.

The virus recognises the four common types of diskette: 360KB, 720KB, 1.2MB and 1.44MB. If the disk's structure does not match any of these, the virus will not infect it. Non-standard (for example, FDformatted) 180KB and 2.88MB floppies are never infected. If the virus recognises the disk type, it writes its own code to the boot sector and overwrites the two last sectors of the floppy with a copy of the original boot sector and the remainder of its code.

Activation

During every boot-up from the hard drive, Finnish Sprayer will check the real-time clock date. If the date is 25 March, the virus will activate, overwriting random sectors on the active partition. The random number is generated by using non-initialised registers as destination values and entering a loop, which calls the BIOS disk write function, decrementing the head value after each write.

After this destructive routine, the virus changes the screen background to grey and displays the text:

```
FINNISH_SPRAYER.1. Send your painting +358-0-
4322019 (FAX), [Aija]
```

Since this text is encrypted with a XOR 50h operation, it is not visible inside the virus code. The phone number is that of the Finnish House of Parliament, which received dozens of faxes on activation day this year. After the display routine, the virus hangs the machine by entering an infinite loop. It should be noted that since real-time clocks are generally available only on AT machines and above, this routine will fail on older machines. On such computers, the virus will never activate. Finnish Sprayer also contains the following unencrypted text, which is never shown:

```
Tks to B.B, Z-VirX ..... [Aija]
```

This string is also used as part of the virus' self-recognition signature. Incidentally, the trigger date of 25 March is also the 'name day' of Aija (a girl's name) in Finland.

Conclusions

The coding style of Finnish Sprayer varies between different parts of the virus. This might indicate that the author has incorporated parts of older viruses into its make-up, although no obvious similarities with other common viruses exist. Another explanation is that this virus might be the work of a number of different people, working as a team.

Finnish anti-virus organisations have followed the Finnish Sprayer incident very closely, which has made it possible to compile remarkably accurate statistics. Some of this information is shown on the map on the facing page.

During March 1994, Finnish Sprayer was reported to have activated on approximately two hundred PCs in Finland alone. The total number of infected machines rises to several hundred, possibly even one thousand. This is quite amazing, since the virus was first found only a few months ago. Such new viruses are becoming increasingly common - for the ill-prepared PC user, the 'writing is on the wall'…

Finnish Sprayer

Aliases:	Aija.
Type:	Memory-resident boot sector virus.
Infection:	Hard disk Master Boot Sectors and floppy boot sectors.
Self-recognition in Memory:	None.
Self-recognition on Disk:	Letters 'Ai' at offset 45h in boot sector.
Hex pattern:	49B8 0103 33DB CD13 0E07 B801 0333 DBB9 0100 B600 CD13 5AC3
Intercepts:	Int 13h for stealth and infection.
Trigger:	Displays message and overwrites part of active partition on 25 March.
Removal:	Under clean system conditions, return original boot sector to its original place. Alternatively, overwrite viral code using the DOS command FDISK/MBR.

VIRUS ANALYSIS 2

Lock up your Source Code!

Eugene Kaspersky

The virus researcher is an indefatigable character, spending his time discovering new algorithms, breaking virus code, and proving time and again that people who write viruses are no match for those who fight to stop them spreading. Nonetheless, every now and then a new virus pops up which poses more than minor problems for the researchers. The latest such innovation is the infection of source code files, and is published as so-called legitimate research by none other than Mark Ludwig.

These viruses are the 'new generation', spreading via a non-executable object. This manner of infection has, of course, always been theoretically possible, but seeing the theory put into practice is fascinating.

Getting There

Early viruses were fairly simple and uncomplicated, spreading by replication - somewhat like cell fission. These creations infected COM, EXE, SYS files, executable files of other platforms such as *MS-Windows* and *OS/2*, and boot sectors. Such viruses alter the code of executable files (parasitic infectors), use substitute file names (companion viruses), or replace boot sectors with virus code. In each case, the virus makes changes to an executable object, or to the way in which an object is treated by the system.

The next 'era', infection of 'pre-executable' files, came with the recent discovery of a virus named Shifting Objectives: this targets object modules, the half-way house between program and source code (see *VB* March 1994 pp.11-12).

Finally, the virus author has turned his attention to the beginning of the chain: the source code file from which the object modules are compiled. Changes are made to the source code, which is then compiled into object modules (OBJ-file) and linked into executable files.

The spread of the source code viruses resembles the reproduction of the butterfly: a larva is dropped, which becomes a caterpillar, which later develops into a butterfly.

Incorporation in Source Code

Source code viruses modify the source code of programs in such a way that once a program is compiled it contains virus code. The sources are known as 'infected sources', and the trojanised executable file as an 'infected file', just as after modification by an ordinary virus. Two problems relate to the spread of such a virus: the first pertains to limitations in the source language, the second to finding the right place in the source code for storage of the virus' source.

In order to infect a source code file, the virus adds its own source code to that file. It is vital that both the virus and the host file are written in the same programming language, because the next stage in the virus' life cycle is at compile time, when the infected file is compiled to form an object module which contains the virus code.

Finding the correct place in which to install the virus source is of paramount importance: if the virus places itself at the wrong position in the infected source, either the compiler will generate error and/or warning messages and will not generate executable files, or the compiled virus code will never receive control from the host code.

Also to be considered is the problem of hiding the virus in the infected source code. If a large block of text is added to the file, even a cursory glance at the file will alert the programmer to its presence.

"the source code virus… hits source code which will then be compiled into object modules and linked into executable files"

Inside the Virus

There are already a number of source code viruses written for the PC (as well as several for *UNIX* - see *VB*, March 94, p.15). The first is written by none other than Mark Ludwig, and begins with the copyright string:

```
/*Microsoft C 7.0-compatible source code virus
This file contains the actual body of the
virus.

This code is (C) 1993 by American Eagle
Publications, Inc. P.O. Box 41401 Tucson, AZ
85717

ALL RIGHTS RESERVED. YOU MAY NOT COPY OR
DISTRIBUTE THIS CODE IN ANY FORM, SOURCE OR
EXECUTABLE, WITHOUT PRIOR WRITTEN PERMISSION
FROM THE PUBLISHER!!! */
```

This in itself is rather an odd statement, given the self-replicating nature of a virus. One assumes that Ludwig (of *American Eagle Publications Inc*) is aware of the fact that it is not yet possible to sue a virus [*Even if it is an artificial life form, capable of evolving! Ed.*].

The remainder of the virus code consists of subroutines which explain the actual workings of the virus, and its infection routine. When an infected program is executed, the virus searches for the string 'INCLUDE=' in the system environment area. Usually, this string points to the subdirec-

tory which contains common header files for the *Microsoft* C compiler. If this is not found, the virus terminates its infection routine and passes control to the host program.

If this environment variable is defined, the virus then creates its own header file, writing its own source code twice to that file, once as the following data array:

```
static  char
virush[]={47,42,77,105,99,114,111,115,111...
```

and also in standard C language:

```
/*Microsoft  C  7.0-compatible  source  code  virus
int  ok_to_attach(char  *fn)
 ...
{
FILE  *host_file;
...
```

This is necessary because executable files do not contain their own source code, but only assembler instructions and data fields. On compilation into an executable file, the source text of the virus will be lost, but the virus must keep a copy of the text in order to infect other source files.

After creating the VIRUS.H header file, the virus searches the current directory for files with the file extension 'C'. If found, the virus scans for the string '#include <virus.h>' to detect already infected source files. If a file does not contain this string, the virus starts its infection routine.

The virus creates a temporary file to read and write infected sources and deletes it after infection. The source file is analysed, and the virus adds the strings '#include <virus.h>' at the file beginning and 'sc_virus();' which calls the virus routine before the last '}' bracket. As a result, the source code file TEST.C

```
main()    {
    printf("Hello  world!");
}
```

becomes

```
#include  "virus.h"
main() {
    printf("Hello  world!");
    sc_virus();
}
```

The virus header file is then written into the directory specified by the INCLUDE variable, and the source code is read through twice: once as a C source for the compiler, and once as a data array. Once these changes have been made, the work is finished, and the .C file is infected. When that file is compiled, linked, and executed, the virus will search for other C files and the entire process will be repeated.

More Source Code Viruses

Ludwig's virus is unfortunately not the only known source code virus. The second is similar but not as long, and contains neither comments nor optimised source code. On infection it creates the file V784.H instead of VIRUS.H, and also uses another subroutine name. The call to the main virus routine is 's784();', rather than (as in the first virus) 'sc_virus();'. Thus, this virus is capable of hiding itself more efficiently than its predecessor.

The last source code virus searches and infects .PAS files, i.e. is a Pascal source infector. It does not create include files but inserts its complete source code into infected files.

It is difficult to fix the length of source code files, as the length of the header file is not the length of the virus (the virus saves its code in header file twice: in hexadecimal ASCII and in source formats). The length of the virus in executable files (and in object modules) depends on the compiler, the compiler version, selected memory model, optimization flags and so on.

The length of the header files in Source_Family_1 is 53256 bytes, and in Source_Family_2, 14955 bytes. The third virus (for Pascal) increases the files by about 52K. The length of executable files grows from about 8K to 20K, depending on the compiler options selected.

Final Note

These three viruses carry their own source code about with them, and therefore do not require disassembly in the traditional way. They are, as discussed, the first viruses to target source files.

The methods and ideas described above are the first from a multitude of possible methods of source code infection. One can only wait and see what crops up next.

Source Family

Aliases:	None known.
Type:	Non memory-resident, parasitic source file infectors.
Infection:	.C or .PAS files, dependent on virus version.
Self-recognition in Memory:	None - the virus does not become memory-resident.
Self-recognition in Files:	File beginning checked for virus source code.
Hex Pattern:	Part of virus source code may be used as search pattern, but it is difficult to detect in compiled files.
Trigger:	No trigger routines.
Removal:	Under clean system conditions identify and replace infected files; delete virus header files.

FEATURE

Heuristics: for Better or for Worse?

When reviewing anti-virus software, it is not uncommon for the main body of the review to be concerned with the virus-specific parts of the product, with other features getting only a cursory mention. Of these overlooked elements, possibly the most interesting is heuristic virus detection. Advocates of the technique point out its ability to provide protection against countless as yet unwritten viruses, while critics claim that the technique is inherently flawed, and prone to false positives. This article examines how heuristic virus detection works, and compares the performance of some of the heuristic-based scanners currently on the market.

Hand-waving or Hard Science?

The underlying concept of heuristics is very simple. An experienced virus researcher could make a good guess as to whether or not a file was infected almost at a glance. This decision would be based on the 'look' of the code. For example, does the file contain code which appears to reduce system memory? Does the program use 'odd' code fragments (e.g. PUSH SP, RET)? A heuristic virus scanner attempts to emulate this 'guess' in software.

True heuristic analysis of computer programs moves quickly into the area of artificial intelligence, but it is possible to illustrate the basic ideas without the discussion becoming too technical. All viruses must contain code to infect other objects, and are usually written in assembler. They very rarely carry out any 'housekeeping' tasks, such as searching for command line options, or clearing the screen, which would be common in a legitimate program. However, they often start with code to make themselves memory-resident, or locate an already resident copy, frequently via an undocumented Int 21h call. Heuristic scanning makes use of these (and other) differences in 'appearance' in order to ascertain whether or not a file is clean.

As an example of basic heuristic analysis, consider a COM file which begins as follows:

```
1233:0100  B405        MOV AH, 05h
1233:0104  CD13        INT 13h
```

This makes a call to the BIOS which would format the fixed disk. Clearly, any program which begins with these instruction is highly suspicious, and is probably a Trojan horse. A heuristic scanner would recognise these instructions as potentially damaging, and warn the user that the file appeared to contain malicious code. Moreover, by using more complex code analysis, routines which attempt to write to other executable files can be located and identified.

Compression and Encryption

One of the areas in which heuristic detection has been seriously limited is examination of an encrypted or compressed file. Evidently, if the main body of a file is not in an immediately executable format, it is impossible to carry out a behavioural analysis of the code.

Fortunately, this problem can be overcome. In the case of compressed files, the developers have added code which decompresses the file 'on the fly', and then subjects the expanded code to heuristic analysis. Thus, any virus lurking inside the compression should be seen.

The case of encrypted viruses is much more difficult to solve, because unlike compressed files, there is no 'standard' virus encryption technique. The approach taken here is to examine the start of the suspect file. If it appears to contain a loop which alters the contents of another memory location, the behaviour of the loop is modelled in software, and the virus code is decrypted. Note that this does not involve actually running the virus, only modelling the changes made during the decryption routine.

Once the loop has completed, the decrypted code can be analysed. This technique allows the scanner to peer within the outer shell of the encryption, and gives excellent results when applied to polymorphic viruses.

Problems

One of the biggest problems with heuristics is that of false positives - i.e. labelling a clean file as infected. This is much more common with heuristics than with a traditional virus scanner. In the latter case, one is searching for something specific; in the former, the software is making an estimate of how much a program 'looks' like a virus. In a large organisation, false positives can be more damaging than false negatives, and vendors have done a great deal of work trying to minimise the problem.

One of the simplest ways in which this can be done is to increase the number of virus-like features a file must have in order to be labelled as infected. Clearly, this reduces the number of false positives, but will also mean that the number of viruses detected by the heuristic elements in the software are reduced.

Another problem is that it is possible to write virus code in such a way that it hides its true function. For example, there are a number of ways in which to hook a particular interrupt: all the virus author has to do is try out several techniques until he discovers one which does not raise any of the heuristic flags. It is very difficult to prevent this type of attack, as the virus author has an essentially infinite time to try out new ideas against the heuristic scanner.

The Tests

In order to test the heuristic components of the products, some very simple tests were carried out. Firstly, the virus scanner was run on a number of infected files, *with all virus-specific elements disabled*. Thus, the figures presented do not reflect the overall detection ability of the software, only that of its heuristic scanner.

Secondly, the software was tested against a large collection of executable files, including a number of files sent to *Virus Bulletin* as suspected viruses which were later identified as false positives. This test-set (100MB of executable files) was used to measure of the number of false positives generated by the software. Details of the virus test-sets used are given in the Technical Details section at the end of the article.

Central Point Anti-Virus

In the Wild	64/109
Standard	142/229
Polymorphic	See text

After the release of *MS-DOS 6*, *Central Point* launched an updated version of its popular *CPAV* product. The new incarnation featured a number of enhancements, one of which was a 'Virus Analyser'. This claims to be an expert system, capable of detecting virus-like code - no further details are given in the documentation.

The results of the tests show that the product is capable of detecting a respectable number of viruses using just its heuristic engine. However, on the test machine, *CPAV* repeatedly aborted with a number of different run-time errors (the following is a typical example: 'WNCPAV caused a General Protection Fault in module <unknown> 24BF:0D42@WWNCPAV will now close'). This problem was repeatable, and associated with scanning particular files. This made it impossible to calculate how many of the polymorphic viruses were detected by the *CPAV* heuristics, but the figure was approximately 550/750.

The report generated by *CPAV* is particularly bland, simply stating that 'Virus Viral Code F' was found in a particular file. According to the manual, this means that the suspicious code was found in a file. In the event of the scanner encountering a boot sector virus, the message 'Viral Code B' is displayed. No further information is given on the *Central Point* expert system, other than stating that the system detects viruses 'using factual knowledge and reasoning'.

The overall detection rate of the scanner could not be measured, as *CPAV* crashed during testing, generating illegal instruction errors. This problem has been reported in *Virus Bulletin* numerous times, and still needs to be fixed.

The product was not wholly immune to false positive problems, labelling one file as infected with 'Viral code F'. With the virus-specific engine enabled, another uninfected file was reported as being infected. When used in a large company with many thousands of machines, this may well be an unacceptably high rate of error.

F-Prot

In the Wild	49/109
Standard	103/229
Polymorphic	39/750

The heuristic elements within *F-Prot* have changed over the course of time, as emphasis shifted from pure virus detection to the realisation that false positives were just as serious as false negatives. Thus, the sensitivity of the heuristics within the program has been reduced. Notwithstanding, the detection results when run against the Polymorphic test-set are rather disappointing.

One of the best features of the heuristic component is that it is possible to receive a report on which virus-like features were exhibited by a program. An example of a typical report is shown below:

```
C:\VIRUSES\INTHEWIL\EDD-2100.EXE  contains
unusual code, which is normally only found in
viruses.

    - self-relocating code

    - invalid time/date

    - strange structure

Please contact Frisk Software International to
check if this is a known false alarm or send
us a copy for analysis.
```

The principal advantage of the *F-Prot* heuristics is that in the false positive tests, the product successfully identified all files as uninfected. This partly accounts for the uninspiring detection results, and makes the product highly useable.

ThunderBYTE Anti-Virus

In the Wild	73/109
Standard	149/229
Polymorphic	594/750

Of all the products tested, the overall winner in terms of detection and usability was *ThunderBYTE*, by *ESaSS*. The detection results were highly impressive, scoring well against each of the test-sets used.

TBAV allows the user to switch between two different levels of heuristics. The default setting is to use minimal code analysis. This is designed to minimise false positives generated by the program, while still detecting viruses missed by the virus-specific search engine. The figures given for the product were taken in this default mode.

The second mode increases the sensitivity of the heuristic engine by lowering the number of features a file needs to have in order to be considered suspicious. With this mode selected, the detection results rose to 98/109 (In the wild), 211/229 (Standard) and 719/750 (Polymorphic). Unfortunately, it also caused a number of false positives: in the low-sensitivity mode *TBAV* had no false positives; in the higher mode, there were 38.

Another feature of note is the amount of information displayed by the product when it encounters a suspicious file. A typical example is displayed below:

```
C:\VIRUSES\INTHEWIL\EDD-2100.EXE  probably
infected by an unknown virus

c   No checksum/recovery information (Anti-
    Vir.Dat)  available.

F   Suspicious file access. Might be able to
    infect a file.

E   Flexible Entry-point. The code seems to be
    designed to be linked on any location
    within an executable file.  Common for
    viruses.

L   The program traps the loading of software.
    Might be a virus that intercepts program
    load to infect the software.

D   Disk write access. The program writes to
    disk without using DOS.

M   Memory resident code. The program might
    stay resident in memory.

T   Incorrect timestamp. Some viruses use this
    to mark infected files.

Z   EXE/COM determination. The program tries
    to check whether a file is a COM or EXE
    file. Viruses need to do this to infect a
    program.

B   Back to entry point. Contains code to re-
    start the program after modifications at
    the entry-point are made. Very usual for
    viruses.
```

ThunderBYTE also deals with encrypted viruses, by analysing the decryption routine and attempting to decrypt the encrypted code. This code is then subjected to further analysis. Using this technique, *TBAV* managed to detect well over 90% of all the polymorphics tested.

Anti-Virus Professional

In the Wild	87/107
Standard	173/229
Polymorphic	577/750

Anti-Virus Professional (*AVP*) is a relatively new name in *Virus Bulletin*, although it will be familiar to subscribers in Russia. Developed by Eugene Kaspersky for *KAMI*, *AVP* is to be found on several anonymous ftp sites in the West.

The sensitivity of the heuristics within *AVP* is not user-definable, and can only be turned on or off. This small limitation aside, the product worked well and detected a large number of viruses. Those detected by the heuristic engine are not described in any detail, beyond the bland statement 'virus Type…' followed by the product's estimate of the virus' length.

Virus detection was extremely good, achieving a slightly lower detection rate than *TBAV*. However, *AVP* did cause three false positives when run against the test collection of clean files - an unacceptably high result.

Analysis and Conclusions

It is rare for any *VB* review to end with a wholly positive note, and this comparative is no exception. However, one pleasing statistic is that, in each case, the heuristic detection was capable of finding at least some viruses. In particular, the excellent detection results of *ThunderBYTE* and *AVP* are worthy of praise.

The largest drawback with a package which utilises heuristic analysis is its propensity for false positives. Although the tests were conducted on far too small a sample of files to be representative, they do show that, on a large site, false positives will happen. For the average user, the high heuristic mode in *ThunderBYTE* is probably unusable, as there is no practical way of ascertaining whether suspicious files are infected or not. A false positive can waste as much time as a genuine virus infection.

These results make it difficult to decide how much of a benefit a heuristic scanner is. The trade off between false-positives and false-negatives is nowhere as clear as for generic virus detection, and the acceptable threshold for error will vary from organisation to organisation. For the dedicated virus hunter, the extensive analysis provided by *TBAV* places it well in front of the competition, but more than a little technical knowledge is required to make the most of its many features. At the other end of the scale, *F-Prot's* heuristics detection results lagged behind those of its competitors, but no false alarms were caused. The ideal choice of product will vary from site to site; it is left up to the reader to decide which (if any) will suit him best.

Technical Details:

Tests were carried out on an *Opus Technologies* 25MHz 386SX, with 10MB RAM, a high-density 3.5-inch drive, a high density 5.25-inch drive and an 80 Mbyte hard drive. The machine was running *MS-DOS 6.2* and *Microsoft Windows 3.1*.

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

[1] **Standard Test-Set:** As printed in *VB*, February 1994, p.23 (file infectors only).

[2] **In the Wild Test-Set:** As printed in *VB*, May 1994, p.22.

[3] **Polymorphic Test-Set:** A special expanded version of the polymorphic test-set, comprising 750 genuine samples of: Coffeeshop (500), Cruncher (25), Uruguay.4 (75), Satanbug (100), SMEG.Pathogen (100).

TUTORIAL

Stealthy Subjects

One of the attributes which is often used to describe a virus is that of 'Stealth'. Unfortunately, the mechanisms of stealth are shrouded in mystery to the average MIS Manager. Much of this image arises from the way in which stealth viruses operate: when a full stealth virus is memory-resident, infected files will appear to be 'clean', right down to the last bit. However, there is nothing magical about this - the trickery described above is nothing more than a simple exercise in assembly-level programming. This article will examine exactly how stealth viruses function, and how their attempts at invisibility can be nullified.

How it Works

The simplest way to understand how one computer program can hide from another on disk is to examine how a typical stealth virus functions. If one were to examine a directory of a disk before and after it was infected with the 4K virus, one would see no changes to file date or time stamps, or to file lengths: indeed, if one opened an infected file and compared it byte by byte with an uninfected original, no difference would be apparent... as long as the virus was already memory-resident on the infected machine.

This invisibility relies upon the way in which DOS programs communicate with the computer hardware.

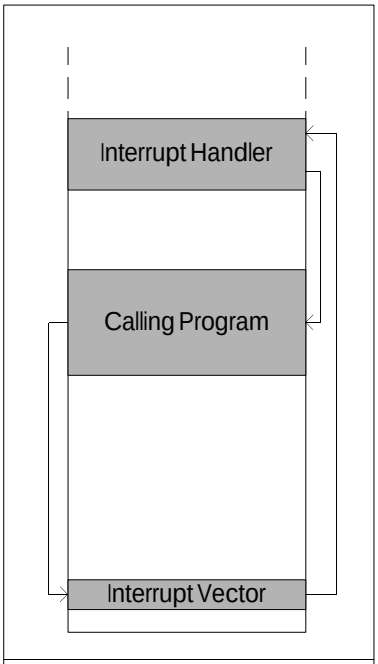
Now You See It...

The purpose of an operating system is to provide an interface between third-party software running on the machine and the low-level I/O functions which nearly all programs need to use (e.g. reading from the disk, outputting to the screen). This interface allows programs to be developed for a wide range of physically different machines with compatibility issues already dealt with by the operating system.

Although a program does not *need* to take advantage of this standard interface, any executable file which accesses the hardware directly would have to be able to deal with the multitude of different standards to which PC components adhere. This would be highly inefficient (not to mention completely impractical).

The interface between a program and the hardware is made up of a number of calls, known as *interrupts*. On the PC, one of the by-products of this design is that nearly all calls to the operating system are made via one or two different interrupts. A precise description of how an interrupt functions on the *IBM PC* is beyond the scope of this article. However, all that is needed to understand how stealth viruses work is a general overview of the actions the processor takes when it encounters an 'INT' call.

An interrupt is an instruction to the processor which instigates a call to another program stored in memory. When an Interrupt (INT) instruction is encountered, the machine makes a note of the current position in the calling program, and then jumps to a subroutine (or interrupt handler) in memory. The location of this subroutine will vary for different machine configurations, so a pointer to it is stored in a fixed area of low memory known as the interrupt vector table. The correct values of these vectors are loaded during the boot process. When called, the interrupt handler carries out whatever function it was designed to do before returning control to the calling program. In the case of DOS, this provides access to all its functions.



Diagrammatic RAM map of interrupt processing on an uninfected PC.

It is relatively easy to write programs in such a way as to be machine independent. Programs communicate with the machine hardware via interrupts, with machine-specific code (e.g. a device driver for an unusual graphics card) dealing with requests to access the hardware. This allows the user to choose from a range of different add-ons which have different characteristics, merely by loading a small device driver which translates the requests of the calling program into instructions which the hardware can understand.

Although there are 256 different interrupts, there are three DOS interrupts which are extensively used by viruses:

- Int 21h - The DOS function dispatcher
- Int 24h - The DOS critical error handler
- Int 2Fh - The DOS multiplex interrupt

In addition to these, there is another commonly used interrupt, Int 13h, which provides direct access to the disk. Using just these four interrupts, a program can access nearly every service needed for basic I/O and functionality. This provides a big advantage in terms of simplicity and usability, but also provides a 'bottleneck' through which calls to the disk usually pass.

Now You Don't!

Due to the nonexistence of memory protection under *MS-DOS*, any program can alter the values stored in the interrupt vector table. This is a double-edged sword. Firstly, it means that a program which wants to intercept calls to DOS (for example, a TSR which monitors suspicious disk activity) can easily insert its own address into the interrupt vector table. However, the downside is that a virus can do exactly the same thing.

A term which is often used within the pages of *Virus Bulletin* is 'interrupt hooking'. This is used to describe the process of altering the address in the interrupt vector table to point to a virus. Thereafter, whenever a program attempts to use that interrupt, control is passed to the virus code. This can carry out whatever actions it deems necessary, before passing control on to the original handler. Armed with this knowledge, one can see that the process of stealth is far from magical - indeed, a rudimentary stealth routine is a simple exercise in student-level computing.

As an example of a such a routine, let us consider the mechanism of hiding an increase in the length of infected files. DOS would normally use an Int 21h call to locate a file. Consider the following code fragment:

```
mov ax, 4E00h          ;Set up Int 21h function
                       ;number
mov cx, 0020h          ;Set up attribute bits
                       ;of file to find
mov dx, _filename_ptr  ;Set DX to point to
                       ;filename
int 21h                ;Call Int 21h handler
```

This issues a call to the DOS function dispatcher to find the first file matching the filename pointed to by DS:DX. On an uninfected system, this would return critical information about the file.

However, if the virus is memory-resident, the Int 21h call passes control to the virus code. At this point, the virus could examine the target file, and attempt to ascertain whether or not it was infected. Techniques for this self-infection check range from examining the time or datestamp of the file for a particular value, down to checking specific code sections within the program.

If this test shows that the virus has already infected the file, it could allow the call to pass to the original Int 21h handler, but with the return address from this call set to the virus code. The virus could then alter the data returned by the DOS call; in this example, it would involve subtracting the length of the virus from the length of the file. Therefore, a directory listing of an infected machine would show no apparent increase in file size.

Extreme Stances

Hiding an increase in file size is perhaps the simplest action which a memory-resident virus can carry out, but several viruses have taken things much further. The 4K virus (aka Frodo.Frodo.4K) intercepts a number of different Int 21h subfunctions. When the virus is resident, it examines nearly twenty different system requests. Aside from the obvious repair of infected file size, 4K also monitors attempts to read an infected file. If such a call is intercepted, the virus returns the original contents of the host file. When that file is closed, the virus examines its extension, and if it is either COM or EXE, infects it.

This means that if one were to use a checksumming package which accessed the disk through the DOS function handlers, no changes would be seen in infected files, and the disk would be reported as clean. However, the action of check-summing the files involves opening and closing every executable, thereby infecting every suitable file on the disk.

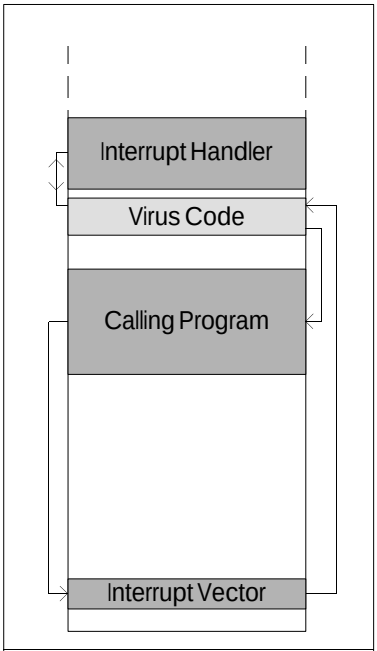
This highlights the need for clean bootstrapping: if a virus is memory-resident, it can block many of the different attempts to search for it. Therefore, many anti-virus packages require that a machine is booted from a clean, write-protected system disk before virus checking takes place.

Fighting Back

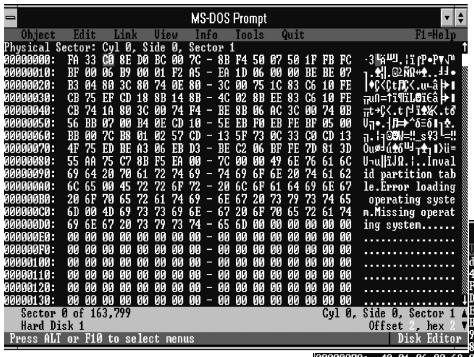
Virus authors are well aware that if their code is not running it is relatively easy to detect. For this reason, they have attempted to improve the stealth capabilities of their creations in a number of different ways.

Firstly, most users still do not carry out a clean boot before scanning a disk for viruses (in the case of a *Windows*-based anti-virus product, this is an almost impossible feat). Therefore, when a scanner is executed, it usually has the option of checking memory for viruses which are known to cause a problem. If any such viruses are found, the user is alerted and requested to clean-boot the machine.

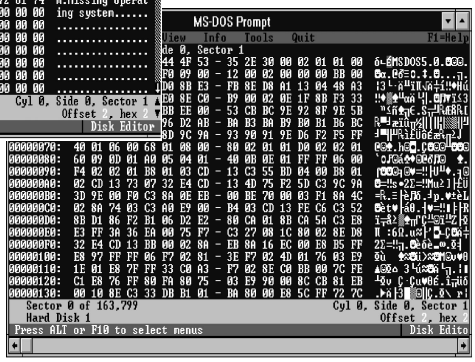
An attack against this technique is to make the virus very difficult to detect in memory by encrypting memory-resident code as well as the virus code on disk (an example of such a virus is Uruguay.6). Finding an encrypted virus in memory is more difficult than finding the same virus on disk: when searching a potentially infected file, it is easy to trace the path of execution



Interrupt processing on an infected machine. The return values of the call are controlled by the virus.



Unfortunately, everything is not as it seems. The results of the same operation are shown below, this time after booting from a clean system disk.



Stealth in Action

Above: the result of viewing a Spanish_Telecom-infected Master Boot Sector with the virus memory-resident. Note that everything appears normal....

through the file, whereas in memory, it is difficult to know where the virus code begins and ends.

An example of a virus which attempts to do this is Satanbug. When resident, this virus multiply encrypts itself, only calling decryption routines when specific areas of the code are accessed. This makes it very difficult to detect reliably once it is running. It is therefore advisable to boot from a clean system disk before checking a disk for viruses.

Another tactic used by the virus authors is to ensure that the virus remains active in memory even after a warm boot of the machine. This trick is relatively simple to accomplish, and means that the user is forced to cold boot his machine by powering off. It is well-worth remembering this fact when examining a machine which might be infected.

New Steps

The latest development in viruses in terms of memory-resident behaviour is that used by the EXEBug virus (see *Virus Bulletin* January 93, p.13). This virus takes advantage of a 'feature' in a particular version of the *AMI* BIOS, which allows the user to force a boot from the C: drive. After the virus code has been loaded in, a boot is 'faked' from the A: drive. Although the user believes that the virus is not active (after all, the machine was apparently cold-booted from an uninfected diskette), the virus is memory-resident, and intercepts all accesses to the Master Boot Sector. Fortunately, the feature in the BIOS on which the virus relies is nonstandard, and therefore does not work on the vast majority of machines. Most anti-virus products will warn the user if this virus is detected in memory, but if a user is particularly concerned about his machine, he should examine the values stored in the CMOS, and check that the drive settings are as they are expected to be.

Things can become still more complicated. Although most viruses implement stealth at an interrupt level, it is possible to intercept disk reads and writes at an even lower level by trapping the calls made to the hardware. Due to the extremely low-level nature of this approach, numerous compatibility issues are raised, but there are viruses which attempt to use this technique.

One example of such a virus is Strange. The virus is a Master Boot Sector infector, which monitors all calls made to the disk at a port level. If it detects any reads of the MBS, it ensures that the original contents of the MBS are returned. The advantage of this technique over the traditional interception of Int 13h calls is that even if an anti-virus program contains 'anti-stealth' code (that is, it attempts to access the disk directly), it will still be 'stealthed' by the virus. Once again, the nonstandard nature of the calls means that Strange will not function correctly on a number of machines, limiting its threat to the user (see *VB*, April 1993, pp.12-13).

Conclusions

Hopefully, it should now be apparent that searching a disk for viruses when a virus is already active in memory is a lost cause, which can lead to even more files on the disk becoming infected. Therefore, when dealing with a potentially infected machine, there are a number of points which the user should take into account.

Firstly the user *must* reboot a machine from a clean write-protected system disk before making any checks for infection. It is essential that such a boot disk is made - the last time at which one wants to be rooting through dusty boxes of diskettes is when a new virus is on the loose.

Creating a boot disk for a 'vanilla' DOS system is easy. However, if any disk compression software is used, it is important to install the appropriate device drivers on the boot disk. This process is discussed in a previous article in *Virus Bulletin* (September 93, p.23): suffice it to say that the quickest way to ensure that a newly-created boot disk is functioning is to test it.

Secondly, the user should bear in mind that a complete power-down of the machine is necessary - a simple Ctrl-Alt-Del is not enough.

The most important fact about stealth viruses is that if they are not memory-resident, they cannot hide the changes they have made to infected files, and the virus author's hard work will have been wasted. Stealth will only work if you let it.

PRODUCT REVIEW 1

Net-Prot: F-Prot for NetWare

Jonathan Burchell

Readers will already be familiar with *Command Software Systems' F-Prot Professional*, which is a well-established package held in high esteem by users and industry alike. This month's review looks at *Net-Prot*, which includes both DOS and NLM versions of the software.

The package seems to set out to be minimalist in its approach, comprising just two 3.5-inch floppy disks and a slim manual, merely ten pages long. No mention is made of the availability of other media - presumably, one could apply to *Command Software* if these are required.

Installation

To install the program, it is first necessary to log in as supervisor. The supplied install program then copies the product's files to the server directory, the user being asked only whether or not AUTOEXEC.NCF should be modified to load *F-Prot* when the server is started. The NLM claims to be compatible with *Novell NetWare v3.11*, *v3.12* and *v4.0x*; however, testing was only carried out under *v3.11*.

When I loaded the NLM, the software immediately generated an error message. The manual points out that *F-Prot* requires the latest version of CLIB.NLM to be installed, and that it may be necessary to download it from a *Novell* distribution site. After some convoluted manoeuvres (the *Internet* is a wonderful thing!) this was achieved, allowing *F-Prot* to load without any error messages.

However, without *Internet* access (or a modem and a *CompuServe* account), procuring the newest libraries is a time-consuming and difficult procedure. *Command Software* should obtain permission from *Novell* to redistribute the latest CLIB.NLM with their software, modifying the install program to copy it over if necessary.

Additionally, why can the install program not check the version of CLIB on the server and warn of any potential problem? This would be far more user-friendly than having the software abort with an error message at the very outset.

Configuration Options

The few configuration options offered by the NLM software are controlled from the setup program, which may be set up to be password-protected. The program can be run by any user who is logged in as supervisor, or who has write permission to the server system directories. It allows for real-time scanning mode, controlling the scanning of files moving on or off the server, and may be set to any combination of file execution, file read and file write.

If a virus is detected, the infected file may be moved to a quarantine directory, or, alternatively, renamed or deleted. *F-Prot* creates the quarantine directory as part of the install procedure, and cannot be set to use a different directory.

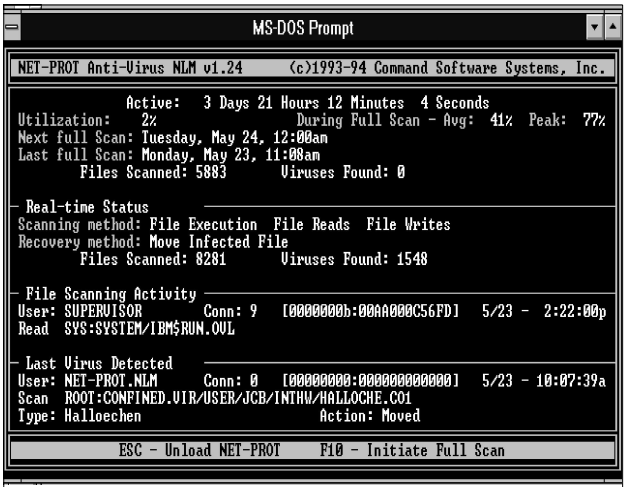
When an infected file is discovered, a warning message may be sent to whomever originated the file request and/or to other nominated users or *Novell* groups. It is not possible to customise the warning message, or to request that it be sent to a group plus a list of individual users.

F-Prot is preconfigured to scan those files most likely to become infected (those with extensions COM, EXE, OVL etc.), but further extensions may be added to the specified list, which appears limited to approximately fifteen entries. This, however, is not enough to retain the predefined list and add all the extensions of the virus test-set. There is no apparent option to specify all files: setting the file extension to a wild card '*' did not work.

The Scheduler which is included is fairly basic, and allows the user to specify scan time according to days of the week and a time to operate. It is also capable of delaying the scan if the file server is in heavy use (qualified as a percentage loading figure).

Points to Ponder

The NLM is very simplistic, and whilst simple can be good, it is necessary to point out the issues it fails to address. First amongst these is the documentation. I am all for conservation, and saving the planet's eco-system: less than three pages in the manual deal with NLM operation and configuration. This would be fine, if good on-line documentation and help existed. *Command Software* provides neither.



Rather surprisingly, given the reputation of the DOS-based product, *Net-Prot* falls down badly in terms of virus detection, missing almost all polymorphic viruses.

The second issue is the fact that the *F-Prot* NLM lacks many of the more advanced features to be found in other server-based anti-virus software. These include the ability to treat groups of servers as logical domains, the ability to load specific NLMs according to scan results, and the ability to customise the entire operation of the scanner. *Net-Prot* does generate a log file, but it is not documented anywhere. No reporting or viewing facilities are provided.

Some very basic features were also absent: I found, for instance, not being able to set the scan to include 'all files' a major disadvantage. Were I to use the software in a production environment, I would want to be able to customise the 'infection found' message. Ensuring that users do not panic at an inappropriate message is of paramount importance.

Furthermore, although *F-Prot* claims to be (and probably is) *NetWare 4.0* compatible, I could see no reference to supporting the more advanced features of *v4.0*, such as automatic data compression and backup media migration.

Neither could I find mention of, nor detect any linkage between, workstation and server utilities. These days one expects the server NLM to act as a centralised logger of events, and to interact with the workstation utilities, double-checking their integrity and ensuring that they are loaded at login time. *F-Prot* has none of these features.

The scanner was not the most efficient I have encountered: about the only thing in its favour is that it seems to place an almost negligible load on the server. I could not measure any degradation of file throughput with the scanner loaded and scanning. It performed poorly on both the Standard and the In the Wild test-sets, and also almost completely missed the polymorphic and *Diet* compressed files.

Within DOS

The *F-Prot Professional for DOS* package provides three main utilities: a DOS-based scanner, a real-time scanner and a file integrity package. A *Windows* program is also installed, which is able to receive and display messages from the DOS TSRs when the user is in *Windows*.

Once installed, *F-Prot* acts as a classic DOS character-based menuing front-end to the scanner, the configuration utilities and the on-line encyclopaedia. The scanner can be configured according to type of scan, disk areas to scan, action to be taken on detection, targets, and files.

Configuration according to type of scan allows a choice of three scanning methods: secure, quick, and heuristic. The secure method uses multiple signatures to detect possible infection, and offers the possibility of disinfection. Quick scan is faster than secure, but not as thorough, and offers no disinfection possibilities. The heuristic scan does not rely on specific signatures, but seeks specific code sequences which may be part of a typical virus. This might be code which writes to the disk via the BIOS, or which attempts to make the program resident in a non-standard way. The advantage

In contrast to the *NetWare* version, *F-Prot* scored well on the test-sets, and was easy and flexible to use.

of the heuristic scan is that it does not rely on the signature database; the disadvantage being that it can be 'triggered' by legitimate programs (see also pp.12-14).

Any combination of local hard disks, diskettes and network drives may be scanned. It is also possible to specify a specific drive/path to be searched. The action to be taken on detection may be set to Report, Disinfect, Delete or Rename. The disinfect and delete actions may be set to be carried out automatically, or to request confirmation from the user.

Choice of targets allows specification of any combination of boot sector viruses, file viruses, user-defined strings and packed files. The files option chooses files to be searched by extension. As well as a standard list, the option of <All files> and a user-specified list are provided. Unlike the NLM-based scanner, wild card extensions are allowed and did work.

On-line Help

In addition to configuring the scanner, the *F-Prot* front-end gives access to a good on-line virus encyclopaedia. This lists all viruses known to *F-Prot*, together with background information on the virus and its actions. An option is provided to define new virus signatures: this prompts for the virus name, the type of infection it generates (boot sector, COM or EXE) and a hexadecimal search string. It is not clear why it should want to know if the virus infects COM or EXE files - what about other file types specified to be searched? Perhaps this information is used to reduce false positives, or is utilised by the heuristic scanner.

The DOS scanner does not have to be used interactively, and may be run from the command line. A large number of option switches allow the run-time configuration and control of the scanner. In this mode, *F-Prot* can be run from within batch files such as AUTOEXEC.BAT. Further control of the scanning process is available: *F-Prot* will generate different DOS ERRORLEVEL exit codes according to scan results.

This scanner has wildly different results to the NLM version, detecting almost 100% of the 'Standard' and the 'In the Wild' test-sets. It was also much better at polymorphic detection, although it did miss all samples of both Uruguay and Cruncher. It is worth noting, that when run in its 'quick' mode, the detection results became identical to those of the server-based software.

The real-time scanner included in *F-Prot* allows checking of files as they are loaded and executed at the workstation. The real-time checker uses the same signature database as the scanner, but will not perform any heuristic-type analysis. If the real-time scanner detects a possible infection, it can be set to notify the user, and to broadcast a network message.

Even though the real-time scanner is a DOS TSR, a user in *MS-Windows* will be successfully notified through the supplied windows utility. I was pleased to see that network broadcasting is supported not only for *Novell* but also for *Banyan* networks. Considering that *Banyan* sites are often enormous, with many users and file servers, this is a useful ability, and one which others would do well to emulate.

The File Integrity Checking Package.

The check program allows checksums to be calculated for files on disk, and calculates new checksums. CHECK.EXE can attempt restoration of altered files which were previously checked. However, the documentation gives few details of the checking method used, or the manner in which restoration is attempted, so it is difficult to assess the integrity of the process.

There is also no indication as to whether or not the checksum database is protected, although it is possible to rename and password-protect it. Presumably, this would provide further security. An additional TSR provides real-time checking of files as they are loaded for execution. It is possible to specify the action to be taken if a file is loaded which is not in the database, or which has been altered since the database was created.

Conclusions

The main problem with *Net-Prot* is simply that the NLM virus detection rate is too low. To be any use at all in day-to-day work, a scanner must score well in virus detection, particularly against those viruses in the wild. This is even more important on a network than under 'vanilla' DOS.

The DOS components were excellent, and I am mystified as to why such a large difference should exist between the two products. The NLM lacks the kinds of features which other server-based products have: its only advantage is that it is extremely simple to install and use. Perhaps if it had a higher detection ratio, it could find a place as a fit-and-forget type scanner/real time checker for small networks. *Net-Prot* is a new product: clearly, further development is needed before it comes into line with the accuracy of the other components of the *F-Prot* stable.

Net-Prot

Detection Results (Secure mode):

NLM Scanner		
Standard Test-Set [1]	205/229	89.5%
In the Wild Test-Set [2]	86/109	78.9%
Polymorphic Test-Set [3]	2/450	0.4%
DOS Scanner		
Standard Test-Set [1]	226/229	98.7%
In the Wild Test-Set [2]	107/109	98.1%
Polymorphic Test-Set [3]	365/450	81.1%

Scanning Speed:

Speed results for an NLM product are inappropriate, due to the multi-tasking nature of the operating system. Full comparative speed results and overheads for all current NLMs will be printed in a forthcoming *VB* review.

Technical Details

Product: *Net-Prot*

Developer: *Frisk Software International*, P.O. Box 7180, 127 Reykjavik, Iceland.
Tel. +354 1 617273, Fax +354 1 617274.

Vendor: *Command Software*, 1061 East Indian Road, Suite 500, Jupiter, Fl 33477, USA.
Tel. +1 407 575 3200, Fax +1 407 555 3026.

Price: £495 per server. Additionally, £89.95 per workstation.

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

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

[1] **Standard Test-Set:** As printed in *VB*, February 1994, p.23 (file infectors only).

[2] **In the Wild Test-Set:** 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, Captain_Trips, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T1, 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

[3] **Polymorphic Test-Set:** The test-set consists of 450 genuine samples of: Coffeeshop (375), Cruncher (25), Uruguay.4 (50).

PRODUCT REVIEW 2

Vi-Spy, with My Little Eye...

Dr Keith Jackson

RG Software's Vi-Spy has been reviewed by *VB* twice before, initially in May 1990, and latterly in August of 1992. In past comparative tests, the product has always performed very well, frequently appearing in the top handful of products. The latest release of *Vi-Spy* adds several new features to the software, which now comprises a scanner, TSR virus detection, an integrity checker, a scheduler, and facilities for removing viruses from infected files. *Vi-Spy* operates under either DOS or *Windows*, and supports a selection of networks. These networking options will be the subject of a later stand-alone review.

Documentation

The documentation shipped with the product comprises two A5 books, identified as a 'Guide to Operations' (154 pages, including a seven-page index), and a 'Computer Virus Primer and Troubleshooting Guide' (67 pages).

The 'Guide to Operations' is very thorough, and easy to understand. In marked contrast to certain packages reviewed recently, all error messages are documented.

The 'Computer Virus Primer and Troubleshooting Guide' describes what viruses are, how to combat them, and what actions to take if a virus is detected. It also provides a good explanation of how a PC boots, and how a virus can interact with this process. Last time around I reviewed this book very favourably: it has not been updated (this would have been unnecessary, as general rules do not alter with time), so the original conclusions still stand.

Installation

Vi-Spy is provided on one 3.5-inch (1.44 MB) floppy disk. Other disks - 5.25-inch (360 KB or 1.2 MB), or 3.5-inch (720 KB) - are available free of charge, simply by filling in a form provided with the documentation. The last time this product was reviewed, the package came on both 3.5-inch and 5.25-inch floppy disks. Although nearly all software products are shipped in this fashion (most notably, *Microsoft*), anti-virus software is needed on every machine in an organisation, right down to single drive XTs. The loss of the choice of media as standard seems like a step backwards.

Installation of *Vi-Spy* onto hard disk was always straightforward: this still holds true. The installation program searches for existing copies of *Vi-Spy*, and decides whether this is an upgrade or an original installation. After performing a fast scan (memory, all boot sectors, and some DOS files), and asking if *Windows* will be used with *Vi-Spy*, any desired subdirectory can be used to contain the 54 *Vi-Spy* files (requiring 1.19 MB of disk space). Changes are optionally made to the DOS file AUTOEXEC.BAT, and the special *Windows* configuration file WIN.INI.

If *Windows* is requested, the installation program 'fires it up', requests that paths to desired subdirectory locations are specified, and leaves the user in *Windows* to test things out. On leaving *Windows*, the DOS installation program re-emerges and completes its tasks - a nice touch.

Scanning

The scanner in this product currently knows about 1879 unique virus names (as opposed to number of viruses). The two previous product reviews searched for 750 (1992) and 46 (1990) viruses respectively - oh, for the simple days!

This scanner is available as either a command line-driven DOS program, a DOS program which uses drop-down menus, or a *Windows* program. All three versions execute the command line version to do the actual disk scan, although it is not possible to alter all of its features from within the menu-driven versions.

> *"Vi-Spy detected all 239 parasitic test viruses, and all nine boot sector test samples"*

When run, all available drives are checked by default, although a scan of a specific drive can be selected on request. Every version of *Vi-Spy* I have seen has contained this feature, which definitely encourages thoroughness.

I tested scanning speed whilst *Vi-Spy* was inspecting the hard disk of my test computer (957 files spread across 36.2 Mbytes), a test which took 28 seconds to complete. In comparison, *Dr Solomon's AVTK* scanned the same hard disk in 21 seconds, and *Sophos' Sweep* took 25 seconds for a quick scan, and 1 minute 13 seconds for a complete scan.

This scan time was measured in what is denoted as 'Optimal' mode, where the product only checks 'important' parts of the disk, and executable files. Although *Vi-Spy* knows about many forms of compressed files, it only warns that such files exist, and does not scan within them.

The scanner can also be used in 'Intense' mode, which inspects many more file extensions (this took 1 minute 18 seconds to scan the same hard disk), and 'Maximum' mode where all files are checked against all virus signatures (increasing the scan time to 3 minutes 10 seconds). A scanning mode entitled 'DOS Critical' is also available: this inspects only the MBS and the DOS boot sector of a hard disk, and completes its tests almost instantaneously.

I re-measured the above timings using the *Windows* version of *Vi-Spy*. When last reviewed, timings under DOS and *Windows* were identical. This is no longer the case: the three measurements described above have increased. An 'Optimal' scan took 1 minute 5 seconds, an 'Intense' scan, 2 minutes 13 seconds, and a 'Maximum' scan (DOS scan 3 minutes 10) needed 5 minutes 18 seconds to complete. These figures represent an overhead ranging from 60 to 130% - clearly something has changed over the years.

The *Windows* version of the scanner is (thankfully) very plain in appearance, and does not have a 'jazzed-up' user interface. This is meant as a compliment - fancy graphics are irrelevant when searching for viruses.

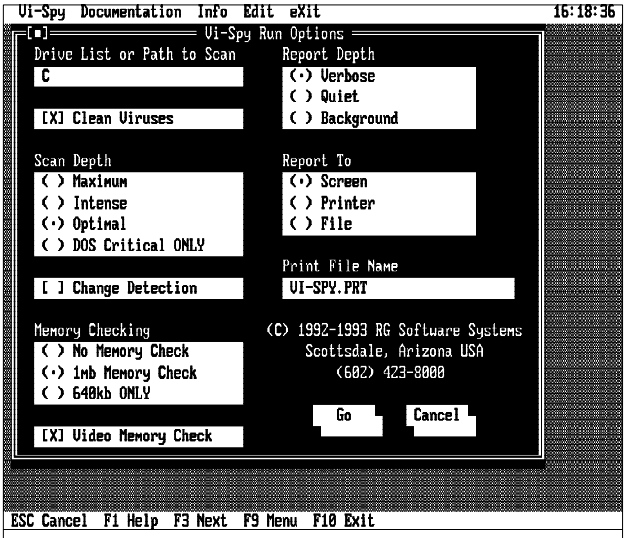
The menu-driven version of *Vi-Spy* and the raw DOS scanner use different naming conventions for various types of scan available (e.g. Optimal = Turbo, and Intense = Full). The manual explains this point, but I find it bewildering, and would contend that it is an unnecessary confusion which should be eradicated.

Vi-Spy has always been very good at detecting viruses, and nothing much has changed in that department. In its last review, it detected all of the test viruses bar one. Even though the number of viruses in the test set has at least trebled since then, *Vi-Spy* detected all 239 parasitic test viruses, and all nine boot sector test samples.

There was, however, a problem when the scanner was tested against the 1024 Mutation Engine (MtE) samples - here, it detected only 83%. Even this, though, is much better than many similar products can achieve.

Memory-resident Features

Three memory-resident detection systems are provided with *Vi-Spy*. The default program (RVS) checks program files when executed, copied, and uncompressed; prevents an accidental floppy disk boot; inspects the boot sector of all floppy disks; warns when a program is attempting to remain memory-resident; prevents writing to a hard disk's Partition table or boot sector, and warns when an executable program has changed in size.



Vi-Spy provides a combination of ease of use and excellent detection, scoring well in all tests.

A second memory-resident program is available with all these features, which additionally verifies a checksum of each executable program before allowing execution. A stripped-down version which only examines the checksum is also available. With one exception, the virus detection capability of the memory-resident component part of *Vi-Spy* proved identical to the virus detection capability offered by the scanner. The memory-resident program spotted all virus samples except one copy of the Liberty virus.

Rather curiously, the other four test samples of this virus were detected correctly, and for reasons I cannot fathom, the memory-resident program decided that one Liberty test sample was not an infected file. Testing the *Vi-Spy* memory-resident program (RVS) against MtE samples produced identical test results to those obtained with the scanner (83% detected), even down to the same MtE test files being missed in both cases.

Small, Speedy and Safe

Any memory-resident monitoring program which is carrying out tests before allowing a file to be executed is bound to have an impact on system performance. This varies from zero (usually from programs which, no matter what they claim, are not actually doing a great deal), up to unusably large overheads.

To test the overhead, I measured the increase in time taken to copy 38 files (1.20 MB) from one subdirectory to another, making sure that the copy was made to/from exactly the same parts of the hard disk. When RVS was not loaded, the test took 17.3 seconds, which increased to 21.5 seconds with RVS active in default setup mode. This is an overhead of 24% on file copying - presumably this also applies to file loading, and is certainly not excessive.

Given that *Virus Bulletin* has published comparative reviews of memory-resident anti-virus products in the past (see *Virus Bulletin*, September 1993, pp.15-19), and been scathing about measured (as opposed to claimed) performance, the above figures are very impressive. Not only does the detection rate of the memory-resident software approach close to 100%, but the overhead of 24% on system throughout is also acceptable.

Most similar products either fail to detect a reasonable number of viruses (in which case they are useless), or impose an unacceptable overhead (in which case they won't get used). Products like RVS are conspicuously rare in the anti-virus industry.

Vi-Spy's memory-resident component can also prevent an accidental boot from a floppy disk. If a floppy disk is left in drive A, *Vi-Spy* intervenes and requests confirmation that a

boot from floppy disk was intended. The memory-resident software occupies less than 17K of memory, and operates in EMS memory, if available.

Checksumming

The first time the integrity checker option is selected, disk scan times increase while it creates a checksum database. For instance, the above-quoted time of 28 seconds to scan the hard disk of my test computer in 'Optimal' mode rose to 34 seconds - a small but noticeable increase. After this first execution, the scanner took only 14 seconds to verify the checksums (when requested).

These quick timings concur with the fact that the documentation states quite clearly that not all parts of a file are checksummed. To keep the checksum verification time within reasonable bounds, this is inevitable.

Calculating checksums across all parts of all files is a very time-consuming process. The details of which parts of a program are included in the checksum process are not explained; however, my tests showed that parts of a file from the 129th byte onwards can be changed at will, and *Vi-Spy* does not seem to notice.

Grouches and Grumbles

I have only a few complaints about *Vi-Spy*, all of which are small. Firstly, the option to maintain a memory map of all memory-resident programs is a pain when a multi-path boot is used. I sometimes use 4DOS as a command processor, and sometimes the *MS-DOS* equivalent, COMMAND.COM. Switching from one to the other causes the boot sequence to stop, and *Vi-Spy* asks if the detected alteration to the memory map is correct, requesting authorisation to update the map. This feature can be disabled, but is present in a default installation.

My multi-path boot also fooled the installation program, which only modified AUTOEXEC.BAT in the final one of the various possible boot sequences. Given that *MS-DOS* now incorporates a multi-path boot as a standard feature from version 6.0 onwards, this needs alteration.

Another grouch is that *Vi-Spy* insists on maintaining its own files in a subdirectory in the root of drive C (called RGVSPYDB). This is a nuisance, and even if it can be circumvented by some jiggery-pokery, *Vi-Spy* should maintain its files within a user-designated subdirectory.

The text file viewer available with the DOS drop-down menu version of the scanner (VSMENU) has an odd quirk. It can be either keyboard- or mouse-driven. However, if one moves down a long way by dragging the text bar with the mouse, when the keyboard is next used, the text file springs back to where the last keyboard command left it. The movement within the file caused by the mouse seems to be ignored. The effect is almost as if the mouse and the keyboard controls are independent of each other.

I was intrigued to see that *Vi-Spy* now includes facilities which will (sometimes?) remove viruses from infected files. Previous versions which I have reviewed did not include such features. While the documentation still states that it is better to delete an infected file and replace it, I think that most users will not follow this good advice, opting instead simply to press a key, and have their worries disappear.

In common with other anti-virus manufacturers, the developers of *Vi-Spy* have no doubt had to bow to user demands for such a feature. I do not have to respond to such commercial pressures, so I shall continue to point out how stupid, and potentially dangerous, such features can be.

However, as the heading of this section suggests, all these observations really are small moans, rather than serious criticism. They are quirks, rather than problems, and given that version 12 of the software has only been shipping for a matter of weeks, they are not wholly surprising.

Conclusions

In its previous incarnations, I have found *Vi-Spy* simple to understand, and easy to use. Additionally, it has always been fleet of foot in searching for virus signatures on a disk. I have found no reason to alter these conclusions in investigating this product for the current review.

The scanner is very good indeed. It is as fast as many of the quickest scanners around, and offers a very high detection rate, though MtE detection needs more work. *Vi-Spy* concentrates on being a very good virus detection utility, ignores the frills, and does not waste its effort on pretty *Windows* front ends, which are ultimately useless in such a package. This one's heartily recommended.

Technical Details

Product: *Vi-Spy*

Developer/Vendor: *RG Software Systems Inc.*, 6900 E.Camelback Rd., #630 Scottsdale, AZ 85251, USA. Tel. +1 602 423 8000, Fax +1 602 423 8389, BBS +1 602 970 6901.

Availability: 8088 processor or better, 256 Kbytes of RAM, 1.5 Mbytes of hard disk space (optional). Either *MS-DOS* or *PC-DOS* can be used. The command line-driven scanner requires *DOS v2.0*; all other *Vi-Spy* programs require *DOS v3.2* or above. Either version 3.0 or 3.1 of *Windows* can be used.

Version evaluated: 12.00, Rel.01.94

Serial number: VSP9412011.

Price: $149.95 with quarterly updates.

Hardware used: A 33 MHz 486 clone with 4 Mbytes of RAM, one 3.5-inch (1.4 Mbyte) floppy disk drive, one 5.25-inch (1.2 Mbyte) floppy disk drive, and a 120 Mbyte 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 1024 viruses generated by the Mutation Engine (which are particularly difficult to detect with certainty).

For a complete list of viruses in the test-sets, see *Virus Bulletin*, February 1994, p.23.

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. 0235 555139, International Tel. +44 235 555139
Fax 0235 559935, International Fax +44 235 559935
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

According to a report in *Corporate Security Digest*, American prosecutors have a poor level of understanding of the technology used in many computer crime cases. Scott Charney, chief of the *Justice Department's Computer Crime Unit*, noted that some prosecutors opt to use more familiar statutes, such as wire fraud, rather than the computer law. 'Computer crime lends itself to other crimes - theft, embezzlement, wire fraud,' commented Jeff Herig, special agent for the *Florida Department of Law Enforcement*. 'If I can prove the easy crime, why go to the trouble of explaining computer crime to the jury?'

S&S International's statutory accounts for the year ending 31 May 1993 show that the company is in the process of being **sued for over half a million pounds** for breach of contract. When questioned, Dr Solomon refused to comment, simply stating that the case 'had nothing to do with failure of the product'.

Network Connection Ltd has launched a product designed to **scan UUENCODED messages for viruses**. The product sits between a protected system and an *Internet* gateway, and checks mail messages for encoded executables. The product uses *McAfee SCAN*, but other DOS-based virus checkers are also supported. Tel. +44 (0)483 776000.

KPMG Management Consulting claims that **79% of company PCs are inadequately protected** against unauthorised access. Brian Kervell-White of *KPMG* commented: 'Though companies now recognise the asset value of the data they hold, until senior management recognise that security is a management issue and become actively involved, the reckless approach to computer security will continue, with all its associated risks.'

Sophos is holding a **Computer Virus Workshop** at the *Sophos* training suite in Abingdon, near Oxford, on 27/28 July. Cost for one day is £295+VAT, and for both days £545+VAT. For further information, contact Karen Richardson. Tel. +44 (0)235 559933.

Jitec Corporation **has announced the launch of a new 'virus-immune computer'**. The anti-virus system is built around the 'totally and literally invincible anti-virus technology, *EVAC* (Electronic Virus Activity Control)'. In the event of this claim being true, it could be the end for over twenty anti-virus software vendors. The industry holds its breath… again.

A team of 600 computer buffs has succeeded in factorising a 129-digit modulus used for data encryption under RSA. However, the group, using 1600 machines, still took eight months to crack the code. This breakthrough poses few problems for current encryption users: most companies and government agencies use 150- or 200-digit keys.

CD-ROM manufacturer *Chinon America Inc* warned users that its name has been put on a **Trojan program entitled CD-IT.ZIP**. The program, which claims to convert an ordinary CD-ROM drive into a CD-Recordable device, destroys critical system files on the user's hard drive. *Chinon* speculates that the vandals picked its company name 'to make it seem that the software was being endorsed by a well-known and reputable CD-ROM manufacturer'. Users are warned not to use the file. Anyone with information which could lead to the arrest and prosecution of those responsible for the CD-IT program are asked to call *Chinon*. Tel. +1 310 533 0274.

The *VB Conference* will be held on 8-9 September 1994, at the Hôtel de France, Jersey. Tel. +44 (0)235 531889.

***VSUM* Certifications for April:** 1. *McAfee Associates ViruScan V114*, 97.8%, 2. *SafetyNet's VirusNet Pro 2.11a*, 96.2%, 3. *Command Software's F-Prot Professional 2.10g*, 96.1%, 4. *Sophos Sweep 2.58*, 93.0%, 5. *Dr Solomon's AVTK 6.60*, 90.5%. NLMs: 1. *McAfee NetShield 1.6V113*, 95.8%, 2. *Sophos Sweep NLM 2.58*, 92.9%, 3. *Dr Solomon's AVTK*, 82.7%, 4. *Command Software's Net-Prot 1.22* 77.8%, 5. *Norton Anti-Virus NLM 1.0*, 76.7%.