ISSN 0956-9979

DECEMBER 1994

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Richard Ford**

Technical Editor: **Fridrik Skulason**

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

CONTENTS

IN THIS ISSUE:

- **A worst-case scenario.** This month sees the sobering news of the shipping of trojanised PC hardware: there is very little which can be done to protect the user against such a threat. Implications are discussed in the Editorial, and a report can be found in the News section (p.3).

- **Taking the limelight.** Natas currently takes ‘pride of place’ as a virus which has quickly spread throughout the world. For an analysis, see p.10.

- **Types of Infection.** The second article in the series on Virus Infection Techniques appears this month. This month’s episode discusses, amongst others, companion viruses and prepending parasitic file infectors.

EDITORIAL

Many a True Word..

Readers of *VB* will doubtless recall the HDZap Trojan (*Virus Bulletin*, May 1994, p.4), and recognise the following short extract from my editorial (*ibid*, p.2):

> *'Not only can the "nasties" be hidden in software, but also in hardware, or the CPU itself. Sooner or later we are forced to place our confidence in something. The Dark Avenger seems to have stopped writing viruses. Maybe he has found a job developing BIOSes for someone...'*

At the time, the comment was not made with any intent at prophecy - a trojanised BIOS, although possible, seemed unlikely. However, many a true word is spoken in jest: only six months later, a trojanised flash BIOS chip with an active payload (see News article, p.3) has been reported. Although *Virus Bulletin* has been unable to obtain a trojanised machine, the precise details of the latest report are immaterial; trojanised hardware has been shipped before: the IBM 4341 contained a logic bomb which halted the machine at 7:30am, 11th April 1980 (John Gaunt, White paper, 'Do vendors deliver?', *TPT* May 1988). Such an attack highlights the danger which may be lurking within an unknown number of computers: examining the contents of the BIOS of one's own machine is a job which no user is likely to undertake.

" it is possible to carry out a surgical strike on ... the central nervous system of a competitor with trojanised hardware "

Imagine selling computer systems with a built-in 'disable' routine which could be remotely accessed... In a world where war is an increasingly high-tech pastime, disabling the IT systems of the enemy is of paramount importance. Without computers churning away, a country can quickly become incapable of functioning. Take, for example, the implications of any company being without its IT system for an extended period of time. Everything previously taken for granted would fall as easily as a row of dominoes, and day-to-day functions would cease.

Many of these ideas have already been realized in fiction (see for example Winn Schwartau's novel *Terminal Compromise*, also available from several *Internet* ftp sites as the file TERMCOMP.ZIP). Happily, the threat is never likely to be as graphic as that dreamed up by the vivid imagination of the author - the current editor of *Virus Bulletin* has yet to be approached by any mysterious temptress, intent on getting him into incriminating positions (should any evil empire wish to try, the Editor may be contacted on Tel. +44 (0)1235 555139 during office hours). Such books should be read as a 'what if...'. However, IT's vulnerability to a concerted, motivated attack should not be underestimated.

This trojanised BIOS (presumably someone's idea of a joke) is the first step along the road to such an attack. Although it is not the first piece of hardware to contain hidden 'value-added' functionality (there is alleged to be a third-party add-on keyboard for the *Apple Macintosh* which occasionally types a message in, if left idle for a certain amount of time), it illustrates a worrying trend in attacks made on computer systems. As more manufacturers enter the computer industry, and as more pieces of 'hardware' are controlled by an EEPROM chip which can be patched at run-time, such attacks will inevitably become more common.

For some time now, the threat posed by an all-out attack on a company's IT systems has been growing. Such a threat is still small (probably even negligible for any one company) but it *is* possible to carry out a surgical strike on what amounts to the central nervous system of a competitor with trojanised hardware. There is no need to design viruses with strong stealth capabilities, and release them on an unsuspecting company's system: just alter the Flash BIOS. On the day of the 'big crash', the perpetrator will be untraceable.

Such predictions will not (one hopes) ever come to fruition. However, the threat is there, and sooner or later, it may be realized. The computing community must be aware of the dangers inherent in such attacks, even if it can do little to prevent them: the possibility of trojanised hardware needs to be recognised. How users and IT managers can hope to defend against its deployment is an altogether more difficult problem.

NEWS

Trojanised PC Motherboard

On Sunday, 13 November 1994, Jakub Kaminski of *Cybec Pty* received what was to be an extremely unusual technical support call. A user had a PC which was playing the tune ‘Happy Birthday’ on every boot, after which the system would hang. Suspecting a new virus, the machine was clean booted… and the problem continued.

Further investigation ruled out the possibility of an EXE_Bug-type attack, so in frustration, Kaminski began to analyse the code stored in the flash BIOS of the machine. At this point, he discovered the Trojan code, which checked the system date, played ‘Happy Birthday’, and hung the machine. Kaminski alerted the anti-virus community without delay; subsequently, a number of other reports of the same problem were received from around the world.

Tracing further information on the trojanised BIOS chips has been difficult, as many suppliers of motherboards have proved uncooperative. It is thought that the boards were shipped from a South Korean manufacturer, and are using an AMI flash BIOS. However, *Virus Bulletin* has been unable to confirm any of these details.

Commenting on the problem, Kaminski said: ‘I think the case shows us how vulnerable the market is to internal sabotage. Even a small loophole in security or quality control can lead to a vast disaster. Treating my pessimism as a matter of principle, I can say that this time the users have been very lucky. It is not hard to imagine a similar situation in which the Trojan did something really nasty. Next year, November 13th is a Monday. It will be quite interesting to see how many users are still unaware of the problem, or at least still suffering from it a year down the track.’

Kaminski would not be drawn on estimating the number of machines affected: ‘Given our experience trying to track down factual information about who is distributing the chip, and the virtually untracked on-sales, we suspect the number could be quite high.’

Providing protection against this type of attack is extremely difficult, and highlights the problems faced by the PC user community. Only time will tell the true extent of the problem, and next November 13th is awaited with interest ▮

Virus Author Charged

In Bergen, Norway, a young virus author has been charged with uploading computer viruses onto a public Bulletin Board System, says the Norwegian newspaper *Bergens Tidende*. The Bergen police force say that it is possible that others might also be charged, but that a lack of familiarity in the area, in addition to restricted resources, is hindering their efforts in the case. Ramus Kjersen, speaking for the Bergen police department, said: ‘Today we have several computer cases under investigation - people who break the law have no reason to feel safe.’ He has high hopes that the case will go to court.

Virus Prevalence Table - October 1994

Virus	Incidents	(%) Reports
Form	23	29.5%
AntiEXE.A	8	10.3%
Stoned	6	7.7%
Parity_Boot	6	7.7%
Spanish_Telecom	5	6.4%
V-Sign	4	5.1%
NYB	3	3.8%
Viresc	3	3.8%
Cascade	2	2.6%
Die_Hard_2	2	2.6%
Monkey	2	2.6%
Aih	1	1.3%
Angelina	1	1.3%
Anti-CMOS	1	1.3%
Athens	1	1.3%
Flip	1	1.3%
Four-on	1	1.3%
HLLC.Fataler	1	1.3%
JackRipper	1	1.3%
Jimi	1	1.3%
Junkie	1	1.3%
Plato	1	1.3%
Stone-o	1	1.3%
Swiss_Army	1	1.3%
Tequila	1	1.3%
Total	78	100%

The report in the *Bergens Tidende* concerning the arrest of the 18-year-old youth, who already had previous convictions for theft of data equipment, appears to have shaken the computer underground in Norway considerably: subsequent to the article being published, several illegal BBSs established in the country were rapidly closed down.

Awareness of the problems which can be caused by computer viruses has been awakened amongst the Norwegian police since the teenager’s arrest. Several cases involving computer crime have already been reported in Norway, but no legal action has been taken until this case arose. A special task force has now been set up at national police headquarters in Oslo for dealing with computer crime ▮

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

Andromeda **CN:** Despite the similarity in name, this 1140-byte virus is not related to Hungarian_Andromeda.

```
Andromeda           8A04 30E4 5030 E03A 0605 0174 0758 FEC4 EBF2 ???? 8AF4 588B
```

Anti-Clerical **CN:** A Polish virus, 877 bytes long.

```
Anti-Clerical       B802 3DCD 2189 84BA 00B9 0000 8BD1 8B9C BA00 B802 42CD 2189
```

Arara **CN:** This is a variable-length, polymorphic virus. It contains the text string '[ARARA]', as well as a long text message, starting 'ILASA MICALAZODA OLAPIRETA' [*Answers on a postcard... Ed.*]. No simple searchstring is possible.

Attitude **CN:** A family of three encrypted viruses; 548, 724 and 825 bytes long. The second searchstring below will detect two variants, and by replacing two bytes with a wildcard, one string will detect all three.

```
Attitude.548        9090 B801 FABA 4559 CD16 E800 005D 81ED 1001 8BC5 051D 0150
Attitude.724/825    9090 B801 FABA 4559 CD16 E800 005D 81ED 0D01 8BC5 051A 0150
```

Aurea **CN:** When the disk-destructive routine of this 653-byte-long virus is activated, it displays the message 'I'm sorry, you lost something because of AUREA'. As the searchstring below is very short, it should be used with care.

```
Aurea               B967 028A 0432 C488 0446 E2F7
```

Australian_Parasite.213 **CN:** Yet another virus from our 'friend' in Australia.

```
Austr_Para.213      B440 B1D5 CD21 B800 422B C92B D2CD 21B4 40B1 03B6 01CD 215A
```

Betaboys.615 **CN:** Also known as 'Rattle', this is a 615-byte virus, probably of Swedish origin.

```
Betaboys.615        5D81 ED03 018D 9E20 018D 968B 013E 8A8E 0301 3BDA 7405 300F
```

Breaking **CN:** A 1000-byte virus, which may display the text 'Your computer is breaking'.

```
Breaking            B800 4CCD 21E8 0000 5B83 EB03 B9E1 03BE 0000 8070 14?? 46E2
```

Carpe_Diem **CN:** The text within the virus (CARPE DIEM! (c) '93 - Raver/Immortal Riot) indicates it is of Swedish origin. The virus is encrypted and 472 bytes long.

```
Carpe_Diem          E800 008B F436 8B2C 81ED 0301 4444 8BC5 0516 0150 EB20 90EB
```

Click **CN:** The name of this 291-byte virus is derived from the string 'The Click' contained within it.

```
Click               B812 3DCD 2172 2089 8515 00B4 3F8D 9512 008B 9D15 00B9 0300
```

Dementia **CN:** An encrypted, 512-byte virus, which contains the text '[DP/1] Dementia Praecox by MnemoniX'.

```
Dementia            E800 005D 81ED 1201 8BF5 81C6 3801 8BDD 81C3 0D01 8A27 8A57
```

Dillinger **CN:** A 547 byte virus. Awaiting analysis.

```
Dillinger           B802 3DCD 211F EB04 EB47 EBCE 50B9 FFFF 8BD8 B43F BABF 01CD
```

Kode4 **CN:** There are two very similar variants of this virus, 282 and 287 bytes long. Both contain the text '-=+ Kode4 +=-, The one and ONLY!'.

```
Kode4.282           B802 3DBA 9E00 CD21 7303 E987 008B D8B4 57B0 00CD 2151 52B8
Kode4.287           B802 3DBA 9E00 CD21 7303 E98C 008B D8B4 57B0 00CD 2151 52B8
```

Kommuna **CN:** This is an 801-byte virus of East European origin. It contains various text strings, such as 'Don't warry please !!!', 'parovozik' and 'KOMMUNA(OSTANOVKA)'.

```
Kommuna             B002 B43D CD21 72E3 8945 E08B 5DE0 8BC7 2D1E 008B D0B9 0300
```

Lapse.323 **CN:** Yet another virus written by the person who calls himself 'Memory Lapse'. It is 323 bytes long and includes the text 'Memory_Lapse.323A (05/28/93) Copyright (c) 1993 Memory Lapse'.

```
Lapse.323           B802 3DCC 93B8 0242 33C9 99CC 2D03 0089 8648 02B4 408D 9603
```

Leprosy.350 — **CN:** A primitive, encrypted overwriting virus.

```
Leprosy.350         0300 EB57 9051 BB3B 018A 2F32 2E03 0188 2F43 81FB 9902 7EF1
```

Mshark.378 — **CN:** This 378-byte Polish virus contains the text ‘Krzemien Limited’.

```
Mshark.378          B802 3D03 D6CD 2173 03E9 8900 8BD8 B43F B904 00BA 1501 03D6
```

Override — **CN:** This virus overwrites the first 1392 bytes of the files it infects, and adds 36 bytes to the end. It is encrypted, and contains the text ‘External Override Dedicated to the jerk who told lies about my girlfriend and almost ruined my life’.

```
Override            E875 01E8 A500 E897 01BB 0500 B970 05B4 40CD 21E8 8A01 E892
```

Panek — **EN:** This 1939-byte virus is of Polish origin, as indicated by a message seen on decryption: ‘Serdeczne gratulacje infekcji’.

```
Panek               2200 A000 0030 0446 81FE 8307 75F7 0413 8AE0 A300 00E9 0802
```

Panic.398 — **CN:** Also known as ‘Just’, this 398-byte virus contains the text string: ‘Don’t panic it ! It is just a virus exist in your system !’.

```
Panic               B802 3D8B D583 C21E CD21 8946 318B D8B4 3FB9 0300 8BD5 83C2
```

PeaceMan — **ER:** This is a 2064-byte virus, also known as ‘Santa’ or ‘Leuk’.

```
PeaceMan            3D00 4B74 1090 9090 80FC FF74 2390 9090 2EFF 2E35 002E 8C1E
```

Phantasm — **CR:** A 366-byte long virus, not yet fully analysed.

```
Phantasm            E2F5 B820 25BB FFFF 8EDB 33D2 CD21 B41A BA80 00CD 215D 5F5E
```

PHB — **CEN:** This virus displays a crude picture when it activates. Two variants are known, one of 4315 bytes (flawed, overwriting) and one of 4461 bytes.

```
PHB.4315            B740 93BA 0001 B9DB 10CD 21B4 3ECD 21EB D7B4 3BBA 0612 CD21
PHB.4461            B440 8D96 9A12 B903 00CD 21B8 0242 2BC9 2BD2 CD21 B440 8D96
```

Piaf — **CER:** 1859 bytes. Awaiting analysis.

```
Piaf                0E1F 0E07 A1E5 07BE 2B01 B97D 068B FE33 DBFE C874 14AC 32C4
```

Pirate — **ER:** The name of this 1344-byte virus is derived from a string it contains.

```
Pirate              80FC 1974 1C80 FC17 7417 80FC 1374 1280 FC1E 740D 80FC 1474
```

Pit — **CN:** This is 611 bytes long and contains the text ‘The Pit v1.20’.

```
Pit                 B801 4333 C9BA FC01 CD21 B441 BAFC 01CD 21B4 4EBA F601 33C9
```

Polifemo — **CN:** This is a 906-byte Italian virus containing the string: ‘**** Polifemo ****’.

```
Polifemo            B43D B002 BA9E 00CD 2172 35A3 3B01 E830 003D 0000 7507 E868
```

Prague.604 — **CR:** This virus belongs to a family that includes viruses which were formerly known as Backtime, Blinker, Joker and Shaker.

```
Prague.604          3D00 4B74 052E FF2E 0E00 5053 5152 1E06 521E 8CC9 8ED9 B824
```

Praying — **CR:** The name of this virus is derived from the string found inside it: ‘Keep On Praying, Jesus’. Two variants are known, 579 and 587 bytes long, both detected with the pattern below:

```
Praying             3D00 4B74 052E FF2E 0D00 9C50 5351 5256 5755 061E 2E89 1611
```

Psychosis.1202 — **CN:** Yet another Swedish virus: at least that is what the following text indicates: ‘Merry Xmas and a happy new year // Sweden - Snowing Again’.

```
Psychosis.1202      BE15 0103 3606 018A 24B9 7304 83C6 3290 8BFE E807 00AC 32C4
```

Rael — **CR:** A 3211-byte polymorphic virus for which no simple searchstring is possible.

Redstar — **CR:** A 352-byte virus that infects files when they are opened.

```
Redstar             E808 0080 FC6C 2EFF 2E94 0050 5351 521E B802 3D9C 80FC 6C2E
```

Rythem.808.A — **EN:** An encrypted, 808-byte overwriting virus. The Rythem family may be related to the Leprosy viruses, but the exact relationship has not yet been determined.

```
Rythem.808.B        BA00 018B 1EE5 0153 E8E0 FF5B B928 03B4 40CD 2153 E8D4 FF5B
```

SillyCN.158 — **CN:** Although the string ‘You’re fucked’, found inside this virus, might indicate that it is destructive, it is in fact quite harmless - it does nothing but replicate.

```
SillyCN.158         B802 3DBA 9E00 CD21 93B9 0500 B43F 8D96 9101 CD21 81BE 9401
```

SillyCN.208 — **CN:** This virus is not really related to the previous one, although they appear to be included in the same family. Just like the HLL* families, the Silly* families are artificial and contain viruses that do nothing but replicate, and exhibit no symptoms that suggest a logical name.

```
SillyCN.208         B802 3DCD 2173 03EB 6690 8BD8 B43F 8BD6 83C2 04B9 0400 CD21
```

SillyCN.215 — **CN:** A simple 215-byte virus that adds itself in front of the files it infects.

```
SillyCN.215         2E8B 1603 01BB DA00 263B 1774 2133 C933 D22E 8B1E 0A01 B800
```

INSIGHT

Hyppönen, that Data Fellow

Megan Palfrey

Like most people in the computer world, Mikko Hyppönen has been around computers almost as long as he can remember. PCs and the Hyppönen family are inextricably intertwined: even before his birth, Hyppönen's mother was working at the *Finland State Computing Centre*. She brought her two sons up in the world of IT, ensuring a computer was always among their toys.

This led to careers in computing for both: the two brothers work for the same company, *Data Fellows*, international publishers and distributors of the renowned and respected *F-Prot Professional.* Mikko is Technical Support Manager, active on the anti-virus side, and his brother is involved with the company's next big project, '*Vineyard*', a *Windows*-based groupware product.

Catching the Bug

Hyppönen went from school to the *Institute of Information Technology* in Helsinki, although he still worked part-time at *Data Fellows*. The company's only contact with the anti-virus scene at that time was in a training capacity. 'Anti-virus products were around, but most companies hadn't started developing them. This was circa 1990; use of this software hadn't become widespread,' said Hyppönen. 'I wasn't interested in viruses at all at that stage.'

Data Fellows' CEO, Risto Siilasmaa, however, *was* thinking about viruses and anti-virus software. Many users were asking the company which product they should buy to protect their system, but no-one had an answer. Products which were available lacked good Finnish technical support, or indeed any Finnish technical support at all. The first step towards solving this was to contact Fridrik Skulason, author of *F-Prot*, and technical editor of *Virus Bulletin*.

'So, the company went anti-virus, but I still wasn't interested. In fact, I couldn't have cared less!' Hyppönen viewed viruses as 'fashionable'; something with which everyone was busying himself. He had no desire to jump on the bandwagon. This attitude was not to last much longer: in late 1990, a virus called Omega appeared in Finland. Hyppönen was hooked.

'I decided to study it,' said Hyppönen. 'It interested me. At that time I knew practically nothing about PC assembly language, which you need to understand to analyse viruses.' So he started to learn, and was soon analysing and disassembling viruses, teaching himself how to extract searchstrings, garnering as much information about viruses and anti-virus software as he could, even dedicating time to research into international marketing of anti-virus products.

Despite this, he sees himself more as a technician than a salesman: 'Most technical people are not marketers. It's the same for me: I don't like marketing hype. I don't like to sell, but it's something I have to do every now and then. Although I don't do much of the actual coding, a lot is done at *Data Fellows*; we create the *Windows* and *OS/2* versions, while Fridrik [Skulason] takes care of the DOS side. I do analyses and co-ordinate the international support. If someone calls with a problem, I try to analyse what's going on. This sort of role reflects what I like to do; check out what's happening, keep up to date in the virus field.'

The World Outside

Viruses are not an all-consuming passion for Hyppönen, however; electronic communication and the *Internet* are other areas of active interest. *Data Fellows* has been developing a *World Wide Web* site relating to virus information which aims to make available to users all the most recent data about viruses held in their laboratories.

He sees advances in technological understanding also in the user community: 'The *Internet* has been around for over ten years now, but until recently there was a wealth of information which nobody who wasn't a computer expert could use. Now, I could take anyone off the street, and if he knows how to use a mouse, he could probably get around the information and forget himself at the terminal for four or five hours, going from one side of the world to another.'

Research and Anti-Research

The issue of contact with virus authors is a complex one for Hyppönen, and problematic for any anti-virus researcher: it is all too easy to destroy one's reputation by being in contact with virus authors while investigating their actions. He views this sort of research as valid nonetheless, albeit a dilemma of the same proportion as whether to buy the now-infamous CD-ROM of viruses released by Mark Ludwig.

There is, affirms Hyppönen, no reason to write a virus, and he believes most people concur with this viewpoint. 'I've created a virus with NuKE's Virus Creation Laboratory (VCL); I believe all researchers have - but why write a real virus? The only reason for an anti-virus researcher to do that would be testing - but it's easy to do that with a program that doesn't self-replicate. Personally, I don't find the idea of writing viruses at all interesting; I've never had the urge, or even thought about doing that.'

The Art of Writing

The topic of virus writers is one with which Hyppönen is familiar, having researched in-depth who is writing viruses and exactly what they are doing. He has been contacted via the *IRC* (*Internet Relay Chat*) by several well-known virus-writers; however, his conclusion was that such contact is not helpful in the long run: ‘It is quite surprising that some virus-writers seem to be very nice persons when you talk to them, but you are always aware that they are doing something wrong, writing viruses. They were probably just bluffing…’

Many people involved in anti-virus research believe the only way to prevent virus authors writing viruses is through re-education, but Hyppönen takes a slightly harder line: ‘Yes, education is important, but they have to take responsibility for what they do. They should be punished - I’m pretty sure that it’s not enough just to tell them that they did the wrong thing and to start to re-educate them. They are breaking the law, or at least acting irresponsibly, and every single one of them knows that.’

He finds it almost incomprehensible that someone could write a virus just for fun, without being fully aware of the implications: ‘I think everybody knows it’s bad.’

There is some small comfort in this story, however: ‘I think that most virus writers just come onto the scene, write a couple of viruses and get out: it’s fairly random. There are very few long-term virus authors - those who make a career of it. Sooner or later, most of them get bored, and leave.’

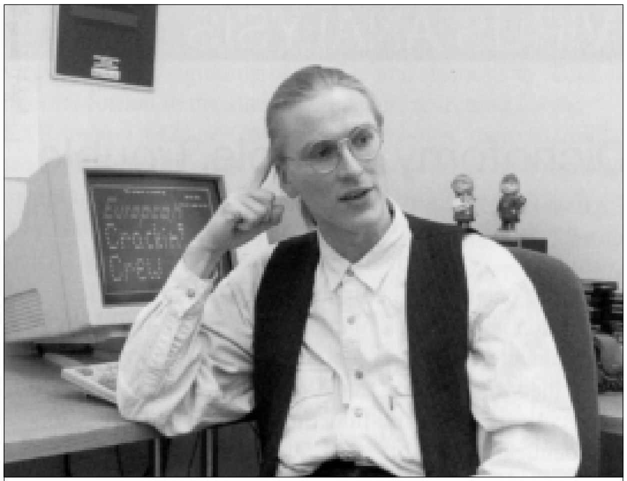
The Shape of Things to Come

It surprises Hyppönen that new products are still appearing on the market: ‘I think most scanners will be overcome by the number of viruses, and I’m not just talking about not being able to keep up with new viruses. I’m talking about practical things, like the fact that products need more disk and memory space; some of them can’t be run from low-density floppies anymore. Products are getting much too big, much too slow. The corporate side will start to look for alternatives, and I believe this will be integrity checking.’

He is convinced that all anti-virus products will eventually incorporate integrity checking; that people will be able to eradicate a virus without knowing what they had. ‘This will become a problem, however, because often, you would still want to know what the virus was; for example, if you had a data diddler on your system, you would definitely need to know that. You can’t be satisfied with generic detection.

‘We will see integrity checkers integrated with scanners that would find only the most important viruses. Virus-specific scanners and integrity checkers will work together, but the scanner will do a much smaller job.’ He sees the industry going towards localization, relying more on local technical support as opposed to a centralized department which may not be able to help regional crises.

‘When a new virus is found in Finland, a competent local support team will be able to find a remedy right away; companies which do not have local representatives will have problems - 24 hours is a long time when your network is down. A week is too long; you start to format your hard drives and go from zero instead of waiting. What companies need now and will need more in the future is local technical support to respond to viruses, and local tools to create a solution to a problem.’



Hyppönen takes a tough line on virus writers: ‘education is important, but they [virus writers] have to take responsibility for what they do. They should be punished…’

As a corollary, Hyppönen sees the number of anti-virus researchers increasing, as the need grows for localized and specialized support. Nevertheless, he does not see viruses themselves as the main threat - the biggest direct risk to a computer is the user himself doing something incorrectly.

‘The problem with viruses will in fact diminish: the more complicated a system, the more different problems it has; for example, reliability. The worst problem with a computer system is that people are so dependent on them that when something does go wrong, they have what you might call a “denial of service” attack.’

Hyppönen’s opinion is that as long as there are programmable computers, there will be viruses; and that viruses are just one more problem with computing in general, to be accepted as a business risk and dealt with accordingly.

The Flip Side

Hyppönen’s life has always been linked with computers: even now, his wife runs her own computer business, specialising in teaching and consultancy - ‘That’s why I married her!’ he joked. ‘We were both workaholics; at the office around the clock, completely computer-minded - in fact, we arranged our first date over a modem. Now we try to separate our work from our private life, though.’

The future is certain: ‘I will stay with viruses; I believe there is a job for anti-virus researchers. If everybody stopped writing viruses now, there would be work for the next ten to fifteen years, anyway. But I don’t see them stopping. And while they’re still writing, I’ll still be disassembling. Where they are, we’ll be right behind them!’

VIRUS ANALYSIS 1

Dichotomy: Double Trouble

Eugene Kaspersky
KAMI Associates

How does one define a computer virus? One possible description is of a block of code which has the property of self-replication, 'infecting' other objects on the system. When the virus spreads, that block is not divided or cut: each replication of a file-infecting virus contains functionally identical executable code. The entire body of the virus is contained in every infected file.

This generalisation may also be applied to boot sector and multipartite viruses. As a rule, all known viruses write their whole code and data on infection. However, as with any rule, there are exceptions. In this case, the exception is the Dichotomy virus.

The virus takes its name from the internal text string '[Dichotomy](c) 1994 Evil Avatar [Dichotomy]'). It is the first virus to use an algorithm which does not place all of its code in every infected file: rather, the virus copies different parts of its code into two different files. When one of these two infected files is executed, the virus becomes active.

Separated Code

The virus code is separated into two blocks, 296 and 567 bytes long respectively. The first section of the code is the virus loader: the virus writes this into files in the standard manner, appending its code to the end of the file, and replacing the first three bytes of the host file with a jump to the attached virus code.

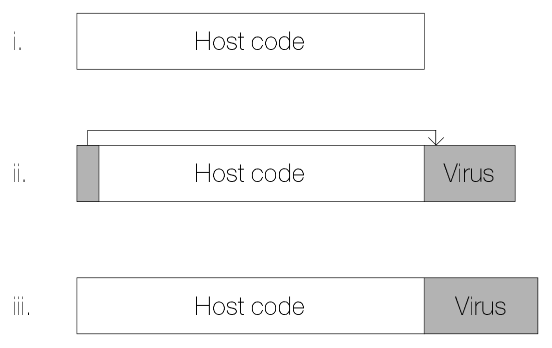


Figure 1: The appearance of the two different types of infected file. i. An uninfected host file. ii. An 'even' file, infected with the loader code. iii. An 'odd' file. Note that the entry point of the file has not been altered.

The second 567-byte block of code contains the remaining virus functions: the code which installs the virus in memory and the Int 21h handler. When files are infected with this second block of code, the virus does not modify the header of the targeted file, but appends that block to the file with no modifications in the original file body. The appearance of the two different types of infected file is shown in Figure 1.

This infection mechanism means that the virus code will not be executed when files infected with the second block of code are executed, as there is no JMP to the virus code. The virus can only be activated when a file infected with the virus' loader code is executed.

Installation

When a file infected with the virus loader is run, control is passed to the virus code appended to it. Processing then transfers to a routine that searches for a predetermined file which has the second part of the virus code attached to it. When such a file is located, the remainder of the virus code is loaded into memory, thus recreating the complete body of the virus in memory.

Dichotomy checks the second part of the code for an identification word, 445Bh or '[D' in ASCII, before continuing installation. This word, located at offset 0352h from the virus' beginning, is taken from an internal text string.

If the identification is positive, an 'Are you there?' call is made. This consists of calling Int 21h with AH=51h (Get_PSP_Address), with ES:BP pointing to the start of the virus code. If there is already a copy of the virus resident in memory, the 'Are you there?' call returns the value FFFFh in the BX register.

Should the 'Are you there?' call go unanswered, control passes to the installation routine, which is located in the second section of the virus body. This routine allocates a block of system memory, copies the virus code into it, modifies an undocumented Memory Control Block area, and hooks the Int 21h vector. The virus then restores the header of the host program, and passes control to it.

Int 21h Handler

The virus hooks Int 21h and checks three of its functions: Get_PSP_Address (AH=51h and AH=62h) and Load_and_Execute (AH=4Bh).

As stated above, Int 21h subfunction 51h is used as the virus' 'Are you there?' call. However, rather than simply checking the value of certain registers, the virus compares the bytes to which the register pair ES:BP points with its own code. If these bytes do not match, the call passes to the original Int 21h handler. This method of checking for an already-resident copy of the virus is very effective, as it avoids the use of a non-standard Int 21h call, making the virus less likely to clash with other software.

The other two intercepted functions are used for infection. The virus has two different infection modes (which I shall label infection of ‘odd’ and ‘even’ files), and toggles between them every time a new file is infected.

If a Load_and_Execute call is intercepted, and the target file is deemed suitable for infection, the virus checks which infection mode it is in. If it is an ‘odd’ file, the resident code appends the second block of virus code to it, and makes no alteration to the file entry point. The name of this file is then patched into a data area in the resident copy of the first block of code.

The next file the virus attempts to infect is classified as ‘even’: the first virus code block (the loader) is copied to the end of the infected file, changing the first instruction of the file so that control is passed to the virus.

> *“Dichotomy ... copies different parts of its code into two different files”*

The virus pays special attention to files located on diskette. To ensure that these files contain the complete virus code, it infects them with both the first and the second blocks. As a result, each infected file contains a complete copy of the virus, just like a standard file infector.

Such double infection, where both parts of the virus are added to the same executable, can also occur if the loader part of the virus cannot locate the filename stored inside it. In this case, the loader issues a call to the memory-resident copy of the virus via the third hooked function - Int 21h, subfunction 62h. This appends the second part of the virus code to the calling file. As with an ‘Are you there?’ call, the memory-resident virus compares the code of the program performing that call: only if the call was made by another copy of the Dichotomy virus is the infection procedure carried out.

After infection, the virus modifies the host file’s date and time stamps. It sets the seconds value to 60 for ‘odd’ files (containing the ‘loader’ block), and to 62 for ‘even’ ones (which contain the ‘installation’ block). Infected files on floppy disks have a 62 second value in their date and time stamps. This stamp is the only form of identification used by the virus to separate infected files from clean ones.

Malfunctions

There are at least two programming bugs in the Dichotomy virus, the first of which occurs on execution of the virus loader. This results in the check of the virus’ identification word being carried out incorrectly.

The second (and most important) bug is the fact that the virus infects EXE as well as COM files. On infection, the virus reads the beginning of the file and attempts to check its internal format in the standard manner, searching for the EXE stamp (‘MZ’ or ‘ZM’ word). However, there is a bug somewhere in the virus code, which appears to be a missing instruction. This results in EXE files being infected as if they were COM files. When such misidentified files are executed, they cause the system to hang.

Several other inadequacies in the virus’ algorithm should also be mentioned. On accessing files, the error flags are not checked as they should be, and file length is not checked correctly: resulting in the corruption of executable files which are very short. Additionally, the virus does not hook the Int 24h vector to prevent the display of DOS error messages when an attempt is made to infect files on a write-protected diskette.

I see this as a new type of experimental virus which can never become prevalent in the wild. In my opinion, the only reason for its conception was to demonstrate just how ‘smart’ the virus-writers can be, and to give an illustration of the ‘dichotomy’ infection technique.

Dichotomy

Aliases: Evil Avatar.

Type: Memory-resident, appending parasitic file infector.

Infection: Any file executed by a Load_and_Execute function.

Self-recognition in Files:

Checks file time stamp for the value 60 or 62 in the seconds field.

Self-recognition in Memory:

An Int 21h call, with AH=51h (Get_PSP_Segment), and ES:BP pointing to the start of the virus code returns FFFFh in BX register.

Hex Pattern: There are patterns for each part of the virus; both can be used to scan system memory.

Part 1:

```
E800 008B DC8B 2F81 ED03 0044
443E 81BE 5203 5B44 B41A 8D96
```

Part 2:

```
FEC4 80FC 4C74 32FE CC80 FC51
740C 80FC 6274 052E FF2E 8C03
```

Intercepts: Int 21h for infection.

Trigger: None.

Removal: Under clean system conditions, identify and replace infected files.

VIRUS ANALYSIS 2

Natas - Fated to be?

Computer viruses, like people, differ greatly in the renown they enjoy. A few, such as Michelangelo, have become household names, even ‘virus superstars’; the majority, however, eke out a miserable existence in virus specialists’ collections. The reason for these varying degrees of fame is not entirely clear… perhaps it is just fate?

Natas is arguably a ‘superstar’ of a virus, with an unlimited target infection area. Thousands of computers have been hit by it, and there are still an unknown number of infected PCs about, notwithstanding the fact that practically all popular anti-virus scanners have been updated to detect it.

Installation

When a Natas-infected file is executed, control passes to the end of the file where the polymorphic decryption routine is located. This code decrypts the body of the virus and passes control to the installation routine.

Natas has features which prevent its code being traced or debugged: the first of these is executed when the installation routine receives control. The virus calls the already-loaded TSR copy and, at the same time, issues an ‘Are you there?’ call (Int 21h: AH=30h, BX=F99Ah).

Due to the virus’ Int 01h handler (see below), it is not possible to trace through the memory-resident routines. Control returns to the host program if the TSR copy is loaded, or if the version of DOS present is under 3.0.

When tracing is in progress, the virus does not make an ‘Are you there?’ call to prevent this happening. Instead it decreases the size of conventional memory as held by the BIOS (at address 0000:0413), cuts the last Memory Control Block (MCB) and copies itself into that freed space.

Installation continues with a tunnelling routine. The virus hooks Int 01h (the Single Step Interrupt), tracing interrupt vectors 13h, 15h, 21h and 40h to calculate their original addresses. The virus’ Int 01h handler is complex, analysing code traced and simulating a ‘no tracing’ state if traced code checks (or appears to check) the trace flag. Using these tracing results, the virus hooks Int 13h and Int 21h by storing five bytes of both interrupt handlers and overwriting the entry point of the code with a FAR JMP to the virus.

Finally, before returning to the host program, the Master Boot Sector (MBS) is infected. The virus saves nine sectors of its body and the original MBS (in encrypted form) in the sectors before the first disk partition (as a rule these sectors are free) and overwrites 29h bytes of the MBS with the virus’ loader code.

One interesting feature of this virus is that the loader code will be placed in different offsets according to the contents of each boot sector. The virus compares the first byte of the boot sector with the JMP NEAR (E9h) and the JMP SHORT (EBh) opcodes and writes the loader’s code into the area to which the JMP instruction points. If there is no JMP, the virus simply overwrites the beginning of the boot sector.

When loading from an infected MBS or floppy boot sector, the virus decreases the size of conventional memory as held by the BIOS, reads its own saved body from disk, hooks Int 13h, decrypts itself, and passes control to the original MBS or boot sector. Natas will infect the MBS on loading from an infected floppy, and on execution of an infected file.

Int 13h Handler

The virus uses the Int 13h handler for three purposes: hooking Int 21h, infecting floppies, and hiding itself on infected disks. Int 21h is hooked in a manner common to most multipartite viruses. On each Int 13h call, the virus checks the address of the Int 21h handler: when DOS installs itself, the Int 21h address changes, and the virus, detecting this at its next call to Int 13h, hooks Int 21h.

The virus intercepts only the Read function (AH=02) of Int 13h. It looks for its identification bytes when reading the infected boot sector (MBS of hard drive or floppy boot sector): if it is infected, the virus decrypts and returns the original code.

Natas will infect any clean floppy boot sector, using a routine similar to hard disk infection. Utilising the BIOS parameter block, the virus finds the last disk sectors, stores its code (nine sectors) there and overwrites the original boot sector code with the virus loader. The remainder of the routine is similar to hard disk infection.

Int 21h Handler

On each call to Int 21h, the virus locks the keyboard by using system ports, and hooks Int 24h. Hooking Int 24h permits the virus to disable the standard error message on writing to write-protected disks. Locking the keyboard is a slightly more subtle trick. Should the computer have any memory-resident behaviour blocker installed, a warning message will be displayed, and the software will wait for a keypress. Thus the virus will make it appear that the behaviour blocker has crashed the system.

Once infection is complete, Natas releases the keyboard lock and returns Int 24h to its original address before passing control to the original Int 21h handler. Natas is also capable of accessing Memory Control Blocks and the System File Table, assisting itself to hide its code in memory.

When resident, Natas intercepts several Int 21h subfunctions: 30h (Get_DOS_Version), 11h, 12h, 4Eh, and 4Fh (Find_First, Find_Next), 42h subfunction 02h (Seek_End), 57h subfunctionns 00h and 01h (Get/Set_File_Time/Date), 3Fh (Read_File), 40h (Write_File), 4Bh (Load_and_Execute) and 3E (Close_File).

Several of the intercepted functions are used to provide comprehensive stealth functionality. The increase in the size of infected files is hidden when they are accessed using the DOS functions Find_First and Find_Next. Additionally, Natas is capable of hiding changes made in an infected file when it is loaded from disk. The virus detects when a write request is made to an already-infected file and disinfects it so the write can be completed without causing file corruption. When the write is completed, the file is reinfected.

Such features are standard in stealth viruses, and suffice for the virus to hide itself well within the system, but sometimes conflict with software such as disk checkers and file compression utilities.

These sort of difficulties are taken care of by the virus: using the Memory Control Block's fields, it checks which program is calling the intercepted Int 21h, disabling stealth routines if that program is CHKDSK*.* (CHKDSK.EXE), AR*.* (ARJ.EXE), LH*.* (LHA.EXE), PK*.* (PKZIP.EXE, PKLITE.EXE and other *PKWare* programs). It also searches for the strings BACK and MODEM in the program names, but I was unable to ascertain to which software these refer.

File infection takes place on calls to Close_File and Load_and_Execute. Natas checks the filename and internal file format in order to determine whether it is in the COM or EXE file format. Secondly, the year field of the date stamp is checked to see if it has had 100 added to the value. This is used as a method of identifying infected files on disk, and prevents multiple infection. When resident, Natas also stealths these small changes to the file date.

The infection procedure is not new from a technical point of view: the virus writes itself at the end of the file and overwrites the file header with a JMP instruction to the virus code in COM files, or changes the entry address in EXE file header. The virus checks COM file length and EXE header fields, and does not infect very small or large COM files, or EXE files with internal overlay code. On infection, the virus encrypts itself with a polymorphic routine, which is new but not as complex as the MtE or the TPE polymorphic engine.

Trigger Routine and Last Notes

The trigger routine is extremely simple and destructive: it formats the entire hard drive. It executes if the virus detects the debugger, or on a one in 512 chance on loading from an infected disk.

The virus contains two internal text strings. The first is 'Natas', which appears after the virus body has been decrypted, and the second is 'BACK MODEM'. The virus keeps the latter string in an encrypted form and decrypts it 'on-the-fly' when comparing it with the file names in the Int 21h handler.

As for variants of Natas, I know of four; 4744, 4746, 4774, 4988 bytes long. They are very similar to the original (the length of which is 4744 bytes), but contain other internal text strings, including, for Natas.4774:

```
 Time has come to pay (c)1994 NEVER-1
```

and for Natas.4988:

```
Yes I know my enemies.
They're the teachers who taught me to fight me
Compromise, conformity, assimilation, submission
Ignorance, hypocrisy, brutality, the elite
All of whitch are American dreams
(c) 1994 by Never-1(Belgium Most Hated)
Sandrine B.
```

Natas

Aliases: None known.

Type: Memory-resident, multipartite, polymorphic stealth virus.

Infection: COM and EXE files, MBS of hard drive; boot sector of floppies.

Self-recognition on Disk:

Compares code at entry (where JMP points) or at sector beginning (if there is no JMP) with bytes E8?? 00BF.

Self-recognition in Files:

100 added to year field of date stamp.

Self-recognition in Memory:

'Are you there?' call with Int 21h (AH=30h, BX=F99Ah). Memory-resident handler returns zero in AX/BX registers.

Hex Pattern: No search pattern is possible in files.

Hard Drive MBS and Floppy Boot:

```
E800 00BF 4000 8EDF 836D D306
8B45 D3B1 0AD3 C88E C0B8 0902
```

Memory:

```
FA2E 8C16 D512 2E89 26D7 120E
17BC D913 FBE8 65FB E421 0C02
```

Intercepts: Int 13h for floppy disk infection, stealth and Int 21h hooking, Int 21h for file infection and stealth.

Trigger: Formats sectors of the hard drive.

Removal: Under clean system conditions, identify and replace infected files, repair infected MBS with FDISK /MBR.

VIRUS ANALYSIS 3

NYB - Grist to the Mill

Derek Karpinski
Andersen Consulting

NYB is a simple boot sector virus with stealth capabilities, which infects the boot sector of diskettes and the partition boot sector of the first hard drive. It consists of a single sector, which is unencrypted and easy to detect. Despite this, it is in the wild in the UK and China.

My overall impression was that this is the product of a person who misread texts on structured programming and safe practice with assembler. Structured programming does not mean that every task must be serviced by an ill-defined function call. NYB would be regarded by most professionals as an accident waiting to happen. Its multiplicity of calls reduces its efficiency, makes detection trivial and, fortunately, makes the virus difficult to maintain or modify. I predict few variants, and suggest that this virus is merely another trivial annoyance.

Action on Booting

Unusually, NYB does not create a stack for its own use. As it makes extensive reference to the stack via the base pointer to store and retrieve data, this may be regarded as a feature liable to cause unexpected results, such as a system hanging.

The virus subtracts 1Kbyte from the BIOS memory available to DOS or subsequent operating systems, copies itself to this area of memory (which is hidden from DOS), stores the original Int 13h diskette handler interrupt, replaces the original Int 13h interrupt, issues a call to the replacement Int 13h handler which will result in infection of a previously uninfected hard disk, loads the original boot sector into memory and continues the boot process normally.

Action when Resident

NYB intercepts all reads from track 0 of all drives. The sector at track 0, sector 1, head 0 is examined for infection; if none is found, a location is specified for storage of the original boot sector. On hard drives this is track 0, sector 11, head 0; on floppies, the last sector of the root directory. As usual, no check is made to see if this sector contains data, so data loss may result. The original boot sector is written to this location, and the virus to the original boot sector.

If track 0, sector 1, head 0 (the location of the boot sector) were read, the call would be stealthed, and the contents of the original boot sector returned instead.

The virus stores the current machine state on the stack shortly after entry to the replacement Int 13h handler, and subsequently accesses the current stack (which may or may not have been accessed by the virus) to determine the machine state on entry - this is done by a series of MOVs. Not good style. Not reliable. But it works most of the time.

Detection and Removal

The virus can be detected through the loss of 1Kbyte of memory after booting: removal should take place following a cold boot from a known clean system diskette.

The SYS command removes the virus from system floppies following a clean boot, but as the original boot sector will still be lurking on the diskette, the user may prefer to FORMAT /S instead. Users of *DOS 3.3* and later may remove the virus from a hard drive with the FDISK /MBR command. On prior versions of DOS, restore the MBS from a known good backup, or examine the sector at track 0, sector 11, head 0, which should be the virus's copy of the original boot sector: this can then be restored.

Conclusion

NYB is a virus which is badly structured, and uses some horrible techniques, but which is still capable of causing infection. It is not the worst written but still functional virus I have encountered - indeed, some areas seem to show some ability - but it is unreliable and unmaintainable.

In short, an uninteresting specimen.

NYB

Aliases: Stoned.I, B1.

Type: Memory-resident boot sector virus with stealth capabilities.

Infection: Master Boot Sector of first hard drive, boot sector of floppy disks.

Self-recognition on Disk:

Compares 40h bytes of virus image from offset 40h with boot sector image.

Hex Pattern:

```
0EE8 AB00 50D1 E8FE CC74 03E9
6C01 5351 5206 5657 1E55 8BEC
```

Intercepts: Int 13h. All reads from track 0, sector 1, head 0 are stealthed, and the original boot sector returned. Reads from track 0 cause the infection routine to activate.

Removal: For hard drives, use FDISK /MBR if supported. For floppy disks use SYS, but preferably FORMAT /s.

FEATURE

Viruses on Pre-formatted Diskettes: LZR revisited

Last month's outbreak of LZR, shipped on pre-formatted diskettes, doubtless provided food for thought for every IT manager. It must now be asked what risks are associated with purchasing diskettes, and how they can be minimised.

Uncertain Parentage

The LZR-infected diskettes were purchased from *PC Superstore*. They were unbranded and pre-formatted, and displayed indication neither of the disks' geographical origins, nor of the company which manufactured them. Such anonymity makes it impossible for a buyer to discover whether quality control checks were made during production, or under what conditions the diskettes were produced.

A representative of a leading disk manufacturer, who was reluctant to be named, stressed his belief that there was a great difference in quality between branded and unbranded diskettes: 'There are a lot of people who say that a disk is a disk is a disk; they're quite simply wrong. Over the last few years the larger manufacturers have done a lot of research aimed at improving diskette reliability and quality. The level of quality between ourselves and some of our Japanese competitors is very, very close indeed, but when you start looking outside Japan, to Asia, the quality differs greatly.'

So, is it worth paying the higher price for a quality diskette? 'Yes. You get a 100% certified disk. Of course, we've upgraded our technology over the last three or four years. We've developed and made numerous changes to the diskette - this is a continual and ongoing process. The most important thing on your PC is the data - you can't take chances with the way you store it.'

Many people in the industry blame the glut of cheap diskettes on a proliferation of small companies in the early nineties. There was a vastly-increased demand for 3.5-inch disks, as companies were abandoning the 5.25-inch diskette, the 8-inch systems, and other storage media.

'There was a big increase in the number of 3.5-inch drives throughout the world, and it was impossible to meet the increase in demand for disks. So, we had to increase our production capacity in several locations throughout the world: despite the fact that we were manufacturing in Japan, the United States and the UK, we still couldn't satisfy demand - it was the same for many manufacturers.'

This led the way to a rapid growth in the numbers of diskette manufacturers, particularly in areas such as mainland China, Taiwan and Hong Kong. The market quickly became saturated, with the result being that prices dropped. 'Subsequently, because of the introduction of anti-dumping duties that were placed in numerous different countries in those areas, what happened was that some of those manufacturers moved their factories to places like Indonesia and

3M: Protecting the customer

One of the principal sources of pre-formatted diskettes is *3M*. The company regards prevention of physical defects and of viruses on the disks it ships as of paramount importance. Paul Gardner, Product Manager (UK) for Diskettes for *3M UK plc*, outlined the steps taken to ensure every diskette is shipped uninfected:

'The [disk] duplicator system is comprised of a master system which controls several duplication loaders (drives). It is a completely separate standalone system, which is not connected to any network, and cannot be accessed by unauthorised personnel.

During formatting, every disk is verified 'bit by bit' against the master program. If a single bit does not match exactly, the disk is rejected.

Every hour, formatted samples are removed from the duplication loaders (drives) for verification on another completely standalone duplicator system. This verifies the diskette 'bit by bit' (i.e. the *IBM* or the *Macintosh* format) against another copy of the master program. The second system has its own master program, so that any differences between the first and second master systems would be detected (e.g. program or formatting defects, viruses).

One formatted sample is removed from each duplicator system at the beginning of each work shift, and verified 'bit by bit': if any 'bit' does not match the master, the disk is rejected.

If a format verification fails during any of the above steps, corrective action procedures are implemented. These include, amongst many special test programs, contacting the appropriate plant personnel, checking the verification system, scanning the sample disks on a PC with the appropriate software. If a virus were substituted somehow for the boot sector, it would be detected during the procedures described above; then, all products formatted since the last verification would be reformatted after the problem was rectified.

Additionally, 10-pack warehouse audit samples are sampled at the end of the packaging line, and the format is verified once again. A virus check is also performed using virus detection programs.'

Malaysia. As far as I am concerned, in order to be able to manufacture disks reliably, you need clean rooms, and you need an audit procedure. The fact that these guys can move overnight says it all.'

Infected Media

Buying from a large well-known name is no guarantee of receiving clean disks, but it allows at least the possibility of redressing any problem which might arise. If you believe your company has received a shipment of infected pre-formatted diskettes, it is important to locate the real source of the virus outbreak, and to take the right steps to prevent valuable evidence being destroyed.

Once a packet of diskettes has been opened and left in a publicly accessible place, it can no longer be said that they were shipped that way. Should they later be scanned and identified as virus-infected, a number of possibilities exist to explain it, the most likely being that the disks have been 'borrowed' and replaced in the box, or that someone has kindly 'checked them for errors' using an infected machine.

In order to minimise any risk, it is important to have set procedures when dealing with all incoming media. The following list is a guideline only, but represents a sensible, effective way of tackling the problem.

- Treat *all* incoming diskettes as potentially infected, and scan them just as you would scan any other incoming magnetic media.
- If a member of staff discovers that a supposedly clean pre-formatted disk is infected, inform the IT department without delay.
- If infected diskettes are found, do not open any other boxes of diskettes shipped in the same consignment: a record of a sealed disk box, opened under controlled conditions and checked on a PC known to be clean will be needed in order to prove that the diskettes were shipped in an infected state. The least likely explanation for infection is that they were part of a large shipment of infected diskettes from a major manufacturer. If you believe you have received a batch of infected diskettes, contact your anti-virus software vendor or *Virus Bulletin.*

Purchasing Decisions

As with all products, there is no absolute reason why a cheaper 'clone' product should be any less reliable than a highly-priced named brand: there is, and always will be, an element of 'paying for the name' in the price tag. Included in the price of the product from a reputable manufacturer is knowledge of the product's origins, and some guarantee that it conforms to certain standards. It is up to the buyer to decide whether this alone is sufficient to justify the higher price of a named brand. Regardless of the source of the diskette, one should adhere to the golden rule that any diskette which is formatted could contain a virus, and should be treated as such.

TUTORIAL

Virus Infection Techniques: Part 2

This is the second instalment of a series aimed towards those readers interested in learning about how viruses function. These articles examine various infection techniques employed by virus writers, and are intended to form a source of reference for anyone involved in the area.

This month, six more infection strategies are considered: companion viruses, path companion viruses, batch file infectors, the ‘Starship’ technique, prepending parasitic file infectors, and EXE Header viruses.

Companion Viruses

It is extremely easy to write a companion virus, as the technique used in infection does not involve altering the target executable in any way. Execution of the virus code is achieved by creating a Trojan file, to which control is passed (rather than to the intended executable object).

On a computer running *MS-DOS*, companion viruses rely for infection on the precedence given by DOS to files with the same name but different extensions. Consider a directory which contains the following files:

```
Volume in drive C is FLIPPER
Volume Serial Number is 0C34-17FA
Directory of C:\PERSONAL\C\TESTDIR

.                   <DIR>   15/11/94  15:05
..                  <DIR>   15/11/94  15:05
MAIN    EXE         10666   14/11/94  18:05
MAIN    COM           286   16/10/94  9:05
MAIN    BAT            19   15/11/94  15:07
        5 file(s)  10971 bytes
                26644480 bytes free
```

If a user types ‘MAIN’ at the command line, which program will be run?

On *MS-DOS* versions of COMMAND.COM up to v6.2, running MAIN without specifying an extension executes the file MAIN.COM. If this were removed, the file MAIN.EXE would be run. Thus, unless one explicitly specifies a file extension, the DOS command line interpreter searches first for a file with the extension COM, then EXE and finally BAT. If no matching file is found, a matching executable is sought in the directories specified in the environment variable PATH. Although it is possible to specify the file extension as well as the name (except in early versions of DOS), this is not a procedure followed by most users.

A companion virus takes advantage of this ‘rank system’ of file extensions in order to place a ‘companion’ file, with extension COM, in the same directory. After this has been effected, if the user attempts to run an infected program, its Trojan companion file is executed. Once the virus has carried out its programmed task, it usually issues an explicit Load_and_Execute call to the original EXE file. The user is unlikely to notice the slight increase in load-time.

In order to hide the presence of the virus code contained in the companion file from the user, most companion viruses mark the extra file with the hidden attribute, making it invisible during normal DOS directory listing.

A virus scanner or an integrity checker will have no special problems detecting companion viruses. In terms of generic detection, a directory which contains two files of the same name, one of which is hidden, would be deemed suspicious.

However, finding two files in a directory with the same name and different executable extensions is not an automatic indication of a companion virus (for example, *MS-DOS v6.22* contains such a pair of files as standard: DOSSHELL.COM and DOSSHELL.EXE).

Only the simplest type of companion file virus has been described above; however, there are many variations on this theme. A somewhat unusual example of the genre, which was first seen in the wild, is Power_Pump: this is a virus that uses a companion COM file to pass control to a master executable, POWER.EXE, which contains the bulk of the virus code.

Another virus which illustrates the companion technique is Carbuncle. This places the Trojan file CARBUNCL.COM in a chosen directory, changes the extension of every EXE file in that directory to CRP, and creates the simple batch file shown below:

```
@ECHO OFF
CARBUNCL
RENAME FILENAME.CRP FILENAME.EXE
FILENAME.EXE
RENAME FILENAME.EXE FILENAME.CRP
CARBUNCL
```

Execution is then passed to the Trojan file, which searches for other uninfected EXE files. When the file terminates, the uninfected host file is renamed and executed.

Given their simple nature, it is surprising that few companion viruses are encountered in the wild. The infection algorithm is extremely simple, and very easy to implement in most high-level languages. One limiting factor may be that they find it difficult to spread from machine to machine, as the hidden companion file is not transferred by the normal DOS COPY command.

Disinfection of companion viruses is trivial, as the ‘host’ executables have not been altered in any way. Removal of the virus’s companion file is therefore all that is necessary.

Path Companion Viruses

A Path companion virus is virtually identical to a normal companion virus, with the exception that the Trojan companion file is not stored in the same directory as the infected executable. Path companions rely on the fact that the user is frequently not in the same directory as commonly used executables. Therefore, when a user types a file name, the operating system searches the areas specified in the PATH environment variable for an executable file of the same name. By inserting either a new search area in the PATH, or by adding the companion file in an area searched before the area in which the host file is located, a Path companion virus, rather than the infected executable, can gain control.

Path companion viruses pose no additional threat to virus scanners. However, as the Trojan companion file and the host file are no longer in the same directory, they are harder to spot using generic detection.

Batch File Infection

In terms of executable objects on the PC, the batch file is by far the simplest, consisting of text instructions to the command line interpreter. However, even the limited functionality of the DOS command line provides sufficient scope for the virus author.

Many viruses which one might at first regard as batch file infectors are in fact companion viruses. However, there are a handful of viruses which are capable of infecting batch files already stored on the disk. The technique used by a batch file infector is more easily understood by showing part of the text of an infected batch file than by explanation. Below is the text found in a file infected by Batman, a typical example of the species:

```
@ECHO OFF
REM <<< binary code: jmp installation, Int_21
handler part 1 >>>
copy %0 b.com>nul
b.com
del b.com
REM <<< binary code: TSR installation, Int_21
handler part 2 >>>
```

The <<< >>> brackets denote those parts of the batch file which consist of non-text bytes.

What is unusual about this particular virus is that its code may be executed either as a COM file or a BAT file: changing the file extension allows the same code to carry out two different tasks.

When the code is executed as a batch file, the virus makes a copy of the batch file with the name b.com, and loads and executes it. The contents of the file are then treated as binary instructions, interpreted as follows:

```
INC AX          ;@
INC BP          ;E
INC BX          ;C
DEC AX          ;H
DEC DI          ;O
AND [BX+46], CL ;<SP>OF
INC SI          ;F
OR  AX, 520A    ;<CR><LF>R
INC BP          ;E
DEC BP          ;M
AND ??,??       ;<SP>
```

These junk instructions do not unduly influence program execution, and once they have been executed, control is passed to the virus code stored in the REM statements.

Batch file viruses present a very low threat on ‘real world’ machines. They are reasonably obvious in their execution, and are trivial to disinfect from an infected machine: it is necessary only to remove the extra lines in the infected file. In terms of detection, they present few problems, either for scanners or checksummers.

Boot Sectors Revisited: Starship

One of the most elegant infection techniques so far encountered is that used by the Starship virus. This method allows the virus to infect a machine by changing merely three bytes of non-executable code.

As discussed in Part 1 of this series, the boot sequence of the *IBM PC* loads and executes the Master Boot Sector (MBS) of the fixed disk. This in turn examines the contents of the Partition Table, in order to ascertain where the active partition boot is located on the disk.

Starship infects the MBS by changing the start of the partition address in the active entry of the Partition Table. This means that the code of the MBS loads and executes the first sector of virus code rather than the original active partition boot sector. Unfortunately, Starship makes no

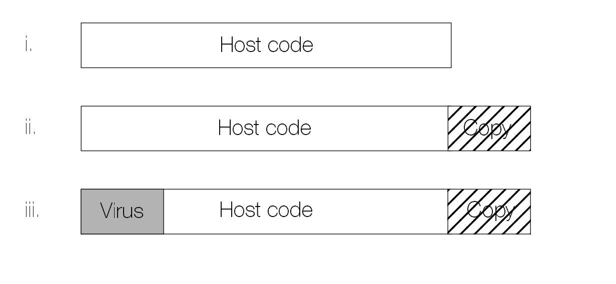


Figure 1: Prepending parasitic infection. i. An uninfected host file. ii. A copy of the start of the target file is appended to the end of file. iii. The start of the file is overwritten by the virus.

attempt to protect the six sectors which it uses to store its own code. However, it allows the virus to infect the disk without changing any executable code in the MBS at all! This technique poses some small problems for detection, as the usual technique for removing MBS infectors will not work: the original partition table must be restored by identifying the virus, or examining the contents of the disk.

Prepending Parasitic File Infectors

Appending parasitic file infectors, which add their code to the end of a host file, and ensure that the code receives control (either by changes to the EXE header or first instruction of a COM file), have already been considered. Code may also be added at the beginning of a file, and a virus which uses this technique is known as a prepending parasitic file infector.

Just like the appending file infector, execution passes immediately to the virus code, which carries out its programmed function, and then restores the original contents of the host file. The scenario is slightly more complicated with a prepending virus, as repair of the memory image involves overwriting its own code. An alternative strategy, therefore, is to load a second copy of the host file, and to disinfect its memory image. This removes the need for the virus to allow an already resident copy to restore the file. Necropolis is an example of a well-written prepending file infector, which is known to be in the wild.

The prepending file infector poses no additional detection problems, either to specific or to generic anti-virus software. As changes are made to the EXE header or the start of a COM file, any sensibly-implemented integrity checker will be able to detect the alterations.

There is a complication during disinfection: when removing an ordinary appending file infector, the start of the host file (usually one or two words) must be repaired, and the extra virus code ‘lopped off’. However, to remove a prepending file infector, much more of the host code will have been overwritten and must be replaced. This can cause problems for certain generic virus removal tools based on backups of the header of a file, and a record of the total file length.

EXE Header Viruses

Virus authors have gone to great lengths in order to hide the changes which they make to files on infection. One trick which is something of a novelty (and only rarely encountered) is to insert the virus code into unused space in the header of an EXE file.

To understand this infection strategy, one must first consider the internal structure of the EXE file header. A diagram of a typical file header is shown below:

EXE file structure:

Byte offset	
0000h	EXE header signature (4Dh or 5Ah)
0001h	EXE header signature (5Ah or 4Dh)
0002h	Length of the file in units of 512
0004h	Size of the file in 512-byte pages
0006h	Relocation table items
0008h	Size of header in paragraphs
000Ah	Minimum number of paragraphs needed above program
000Ch	Maximum number of paragraphs desired
000Eh	Segment displacement of stack
0010h	Contents of SP register at load time
0012h	Checksum (word)
0014h	Initial value of IP register
0016h	Segment displacement of CS
0018h	Offset of first relocation item
001Ah	Overlay number
001Bh	Reserved Relocation table Reserved Code and Data

The length of the EXE file header is always an integer multiple of 512. Therefore, a simple EXE file which does not contain a large number of relocatable items frequently has a large amount of unused space in the header, and this is filled with zeros.

An EXE header file infector takes advantage of this unused space by overwriting the zeros with its own code, and altering the header so that execution is passed to the virus’ entry point. The principle ‘advantage’ of this technique is that the virus does not increase the length of the infected file, and does not need to be memory-resident. The main disadvantage, however, is that the virus’ size is limited to a maximum of approximately 480 bytes.

In terms of detection, and disinfection, no particular problems are posed by this infection algorithm. Due to the small amount of virus code concerned, viruses which use this type of infection mechanism tend to be relatively simple.

The Next Episode…

The next instalment in this series will examine some of the more uncommon techniques, such as that used by Commander Bomber.

PRODUCT REVIEW 1

InocuLAN: Server Protection

Jonathan Burchell

Cheyenne Software has a long-standing record, well-known to many, for producing reliable *NetWare*-based software. Indeed, it would not be an exaggeration to say that most network administrators have at some point in their career used, or at least seen, *Cheyenne's* server and workstation data backup utilities, *ARCserve* and *ARCsolo*.

The company is hoping that their anti-virus product, *InocuLAN*, will also gain such an enviable reputation: this review takes an in-depth look at the latest release, version 3.0, which represents a major update to the product, adding more functionality as well as improved virus detection.

In the Box

InocuLAN comes in an attractive clamshell-design box containing two manuals (Supervisor and User) and seven 3.5-inch, 1.44 MB disks: 5.25-inch disks are not included, but are obtainable from the manufacturers. The ALERT NLM, with extensive server-based messaging features, the *InocuLAN* Server NLM (anti-virus package), and *MS-DOS*- and *MS-Windows*-based tools to administer the servers (with scanners and TSR to install on workstation) form the larger part of the package. Also included are GETBBS (an NLM which automates downloading of *Cheyenne* BBS signature files in real-time), a DOS-based virus encyclopædia, and client *Mac* software for workstation protection.

InocuLAN works on *NetWare v3.11* and above: support for earlier *NetWare* versions has been dropped. *NetWare 4.0* systems are supported if bindery emulation is enabled, but no specific support of advanced 4.0 features (e.g. data migration and compression) is provided. CLIB must be at least version 3.11d.

Installation

The first component to be installed is the ALERT NLM, using an installation routine on disk. Next, the *InocuLAN* Server NLM must be installed: this procedure too is mostly automated. One question which needs to be answered about server installation is whether the system login script should be modified to run AVUPDATE automatically when a user logs in: at the moment, this is optional. The program updates workstation executables and signature databases, ensuring that every user is running a current version of the software. It also permits controlled automatic distribution of signature files, whether or not a server is part of a domain.

Once the NLMs are in place, the *InocuLAN* Manager, which includes DOS and *Windows* interfaces, and the anti-virus TSR, Immune, may be installed. A full installation needs nearly 10MB of local disk space. The install program can also create a rescue floppy disk containing the *Cheyenne*-defined 'Critical Disk Areas', which consists of the Master Boot Sector (MBS), the partition table, IO.SYS, DOS.SYS, the CMOS memory, and the current shell program (this is COMMAND.COM in most systems).

The *Windows* installation program installs *Windows* and DOS interfaces. The DOS installation program is similar, but installs only the DOS software, and can be driven entirely from the command line. All workstation functionality can be invoked via the command line with server login script. The DOS installation program can install the DOS software automatically on all workstations as they log on to the network - a technique which could save considerable time at a large site.

The protection TSR is identical in both the *Windows* and the DOS versions of the product. It is, therefore, possible to automate completely the task of ensuring that each workstation runs the base level of *InocuLAN* protection software.

An electronic encyclopædia, VBASE, comes with the product: oddly, it is not included in the installation process, nor mentioned in the manual. This is a shame, as it is extremely instructive, and provides a wealth of information about viruses, virus authors and related subjects, including a number of recovery techniques. VBASE is not developed or maintained by Cheyenne; it is licensed from *Norman Data Defense* (of *Firebreak* fame).

Administration Features

Both DOS- and *Windows*-based versions of the software provide similar administration facilities: the former uses a character-based menu system; the latter takes full advantage of the *Windows* GUI. *InocuLAN* is, I think, unique in that in addition to providing DOS and *Windows* administration tools, most aspects of server operation (excepting scheduled scans) can be configured from the server console itself.

At a site with many servers, those which have the Server NLM installed may be grouped into logical administration units known as domains. Each group has a nominated master server, and changes in configuration, scheduled or immediate scanning jobs, or updates to signature files are shared between it and other servers belonging to that domain.

Both versions of the administration program provide similar features apart from domain administration, including real-time monitoring, scheduled/immediate scanning, an activity log, a reports log, and a scan report file.

InocuLAN's enforcement feature can detect if a workstation is running its TSR, Immune, and can force that workstation off the network if Immune is not present. Unlike several other products which include this feature, *Cheyenne* has added the concept of a grace period: a user not running this component is notified, and given a specified time to load it. If he fails to comply, he is forcibly logged off the server when the grace period expires. Enforcement can be controlled down to the user and group level.

Server Scanning

The Server NLM can be set to scan DOS files and, optionally, *Mac* files. All files, or all executables, may be scanned. Note that the detection results in the table on page 20 were obtained with *InocuLAN* in its 'all executables' mode. For DOS files, an executable is defined by a list of extensions which can be modified by the administrator - the default list is APP, COM, DRV, EXE, OVL, OVR, PRG, and SYS. *Mac* files may be defined as all files, application type, or those with a resource fork. The real-time monitor may be set to track incoming, outgoing, or incoming *and* outgoing files.

Unlike program files, data files can be scanned in two different modes: Fast Scan, which checks the beginning and end of a file for virus code, and Full Scan, which checks the entire file. Various measures are available on detecting an infected file: take no action but send ALERT messages, delete the file, rename the file, purge the file (i.e. delete it, and ensure it cannot be recovered later through the use of utilities such as Salvage), move the file to a user-specified quarantine directory, and move and rename the file. Any actions are recorded in that server's (and master's) audit log.

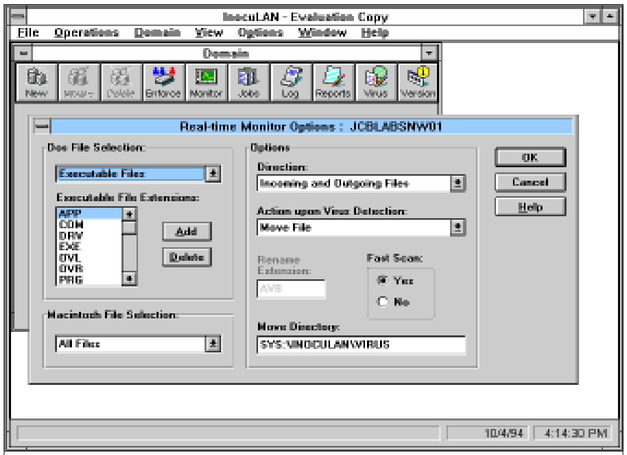
InocuLAN maintains a list of scheduled scans. To set a scan, the volume and/or directory to be scanned must be specified, as well as the CPU utilisation threshold at which the NLM should slow down. The default time to start the scan is immediately, creating an 'on-the-spot' scan, and the optional repeat interval is months, days, hours or minutes.

The other options for scheduled scans are similar to real-time scanning in terms of specifying which DOS and *Mac* files to scan, and what to do on detection. One option, 'Cure File', attempts to repair an infected file automatically. *VB's* views on file repair are well known - the Server NLM attempting such action almost 'on the fly' could well be the worst of all options. *Cheyenne* seems equally sceptical, and does not offer the option at all in real time. Indeed, the manual states, 'Even if InocuLAN cures the file, we recommend you purge the infected file and restore the original'.

Scans can be further refined by defining a list of files and directories to be excluded from the scan. It is also possible to instruct the results of the job to be dispatched to a specified print queue automatically.

Logging and Reporting

InocuLAN maintains an activity file on the server or master server which reports on NLM activity, viruses discovered by Immune or the server real-time monitor (workstation scans or scheduled server scans have their own report files), server signature file and configuration changes with operational results. The log file is comprehensive, reflecting most operations. Simple file searching is provided but no sophisticated filtering, reporting or printing utilities are offered.



Real-time Monitor Options : JCBLABSNW01

Dos File Selection:

Executable Files

Executable File Extensions:

APP
COM
DRV
EXE
OVL
OVR
PRG

Macintosh File Selection:

All Files

Options

Direction:

Incoming and Outgoing Files

Action upon Virus Detection:

Move File

Rename Extension:

AVB

Fast Scan:

● Yes
○ No

Move Directory:

SYS:\INOCULAN\VIRUS

Cheyenne's InocuLAN provides an excellent combination of user interface and reliable detection results.

Scan results are stored in a reports file (one per scan job) which incorporates a basic search and a simple print facility. Like the activity log, this is a plain ASCII file which gets very big, very quickly. Although data in both files can be deleted, it would be more efficient to have sophisticated filtering and reporting tools. As neither file is documented, writing one's own report generator may be difficult.

Configuration

I was surprised that the workstation interface lacks the ability to view the status screen of the NLM. This screen shows current NLM status and scan progress, and can only be viewed on the server console or via rconsole.

The ALERT NLM can only be configured via its *NetWare* console interface, either directly or using rconsole. This is regrettable, as it is a powerful, flexible component, capable of sending messages about a virus detection to bindery-specified users via a *NetWare* broadcast mechanism, and to electronic pagers and fax machines via a modem attached to the server. ALERT can also send Email via *Novell* MHS messages, printed messages to a print queue, or SNMP messages to SNMP-based network management stations.

Workstation Software

The GUI front-end gives access to the workstation scanner and the domain scanner. The stand-alone scanner (supplied in DOS and *Windows* format) offers similar features to the scheduled scanner in the *NetWare* software, including the ability to cure infected files. Unlike the *NetWare* scanner, it can scan local and mapped drives and boot sectors.

The critical area backup program (defined in the Installation section) can save and restore the critical areas. The software backs these areas up to a floppy (creating a rescue diskette) or to a network server. When writing to a server, a unique filename, based on the workstation's hardware network address, is generated for the backup data. Additional functions allow for backups to be restored, and for a comparison to be made between the current version of this data and a previous backup.

The Examine utility checks for changes to the critical areas, and is designed to be run at AUTOEXEC.BAT time or from within the login script. Little information is provided as to how checks are made, although I did notice some SIG files in *InocuLAN's* directory which appeared to be a simple backup of the information. If no checksum data exists elsewhere, however, this would be an easy system to target.

The TSR component, Immune, claims to provide real-time protection against viruses using signature analysis and virus behaviour detection. The manual does not document which types of viral activity the TSR can detect, and although I did not investigate that feature, I can report that with Immune loaded and the activity features enabled, I was able to lead COMMAND.COM into debug, modify it and write back to disk without Immune objecting. Perhaps *Cheyenne* regards such action as lunacy rather than viral activity!

Immune provides further workstation protection by scanning files as they are copied or loaded. Where expanded or extended memory is available, the TSR will consume about 7K of conventional base memory and approximately 250KB of extended memory. There are three versions of Immune available; small, medium, and large, each providing differing compromises between memory footprint and detection accuracy and speed.

Conclusions

All components of *InocuLAN* perform well: in fact, the TSR results are amongst the best I have seen for a component reliant on its own file checking. The detection results as a whole are impressive; not quite good enough to place it out in front, but certainly a healthy second or third.

The principal weaknesses in detection lie in the missed Uruguay and Cruncher infections. Although *InocuLAN* found most viruses in the 'Standard' and the 'In the Wild' test-sets, not all of the latter were found. There is no excuse for not being able to find all viruses in the wild; it is to be hoped that newer signature databases will improve on this.

The most remarkable feature of *InocuLAN*, in my view, is that it combines excellent detection with full GUI and domain management. It has the versatility and slickness of products like *Norton* and *Central Point*, combined with a detection ratio which could easily be improved to 100%; the prime consideration in any anti-virus package.

InocuLAN's range of facilities, and its interface quality, is superior to many 'better' detectors which are available; a factor which could well prove important on a large multi-server site. If detection ratios improve, this product could challenge any on the market today. One to keep monitoring.

InocuLAN

Detection Results:

NLM Scanner:		
Real time		
Standard Test-Set[1]	219/229	95.6%
In the Wild Test-Set[2]	99/109	90.8%
Polymorphic Test-Set[3]	501/600	83.5%
Background scan		
Standard Test-Set[1]	221/229	96.5%
In the Wild Test-Set[2]	100/109	91.7%
Polymorphic Test-Set[3]	502/600	83.7%
Workstation Scanner:		
Standard Test-Set[1]	223/229	97.4%
In the Wild Test-Set[2]	101/109	92.7%
Polymorphic Test-Set[3]	500/600	83.3%
Immune:		
Standard Test-Set[1]	223/229	97.4%
In the Wild Test-Set[2]	102/109	93.6%
Polymorphic Test-Set[3]	502/600	83.7%

Technical Details

Product: *InocuLAN v3.0.*

Developer: *Cheyenne Software*, 3 Expressway Plaza, Roslyn Heights, NY, USA. Tel. +1 516 484 5110, Fax +1 516 484 3493.

Price: Including work, home, and remote client use - US$395 for 25 users, US$595/100, US$795/250, US$995/1000. Sold in server-sized site licences; new signatures databases (included in price) can be downloaded automatically.

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

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

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

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

PRODUCT REVIEW 2

InVircible: InVincible?

Dr Keith Jackson

InVircible, 'The World's most effective anti-virus system', claim its vendors. 'InVircible, The Ultimate Anti Virus Protection', says a file on the product's master disk. Pretty tall claims to live up to.

This product consists of a scanner, several 'repair utilities', and an integrity checker, which claims to be able to detect known and unknown viruses. It also offers network capabilities and operation under *OS/2*: as this review concentrates on the DOS software, these features were not tested. The integrity checker and the scanner include features which purport to be able to remove viruses from infected files.

Documentation

The manual, an unbound, unindexed, 46-page A5 booklet, provides a good description of the theory behind *InVircible*, and an adequate explanation of how to use its individual components. Readability, however, is not helped by the fact that pages 28A and 28B are simply stuffed in between pages 28 and 29, with no attempt made to maintain continuity.

The documentation is prone to making claims which are palpably untrue. For instance, its scanner is claimed to be 'faster, safer and more efficient than any other on the market'. *InVircible* is indeed fast at scanning, but certainly not the fastest; *ThunderBYTE* (to name but one competitor) beats it hands down. The scanner's efficiency at virus detection is also noticeably poor (see measurements below).

Scanners in general are rubbished in the documentation, in such phrases as: 'Polymorphic viruses have rendered scanners effectively useless since they cannot be removed by an algorithmic approach'. This is, of course, untrue. The manual also contains a two-page diatribe against memory-resident components, which, despite some salient points, spoils its arguments through over-emphasis.

Copy Protection

Regular readers of these articles will know that *VB* does not review copy-protected products, taking the stance that such products breach a fundamental rule of security; i.e. that accurate and plentiful backups of all disks should be maintained. The introduction to *InVircible's* manual states that the product is copy-protected: when asked, the vendors, *New Castle International*, denied this, describing the process as 'registration or personalisation'.

The company claims that this is similar to procedures used by products such as *QEMM* and *Stacker*, both of which require user registration information to be written back to the floppy disk used for installation. They omit to say that it is possible to make as many backup copies as desired of *QEMM* and *Stacker* floppy disks, and to install from these backups, unlike *InVircible.* Further, the developers claim that this scheme is 'favored by corporate and institutional users'. If this is true, why pretend it is not copy-protected?

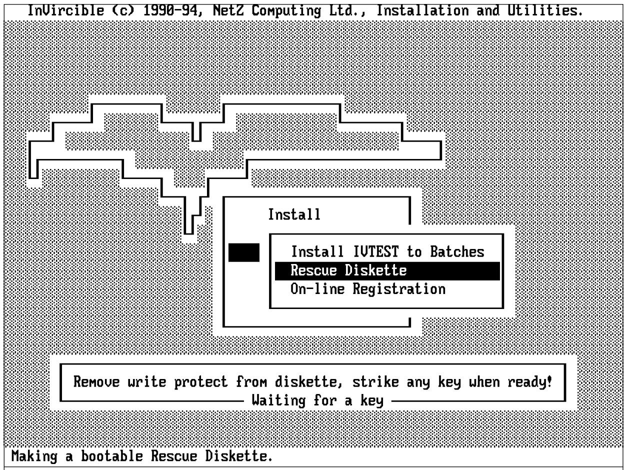
If *InVircible* is installed from a copy of the original floppy made using DISKCOPY, not all its features are available, despite the fact that DISKCOMP thinks the original floppy disk and the copy are identical - restoration functions are disabled. It is thus not possible to take a *complete* backup copy of the floppy: wherever this is the case, a disk is copy-protected. The rest is marketing fog, designed to confuse.

Installation

The installation process creates its own directory on drive C, then scans for viruses, copies the required files, and creates a set of 'Integrity Signatures'. Two lines are inserted at the beginning of AUTOEXEC.BAT, which verify the PC's integrity before other programs are allowed to execute. What about viruses which may be inside already installed EXE files as device drivers when AUTOEXEC.BAT is executed? Installation also produced a series of high-pitched squeaking noises whilst *InVircible* files were being copied. Most odd.

When the installation process was complete, *InVircible* had added itself to the *MS-DOS* PATH, and five files to drive C's root directory. Such file scattering is unforgivable. The product's report files are also created in the root directory, rather than in its own, which would be far more sensible.

After everything was installed, a message appeared onscreen saying: 'Prepare the Rescue Diskette immediately after rebooting'. Rebooting is requested, not enforced - nothing onscreen warns users to remove the master disk first. The



```
InVircible (c) 1990-94, NetZ Computing Ltd., Installation and Utilities.
```

```
Remove write protect from diskette, strike any key when ready!
Waiting for a key
Making a bootable Rescue Diskette.
```

Invircible uses a number of techniques in an attempt to provide generic detection of viruses.

rescue disk is set up as a bootable floppy, and information about the partition, the boot sector, and file integrity is copied across, as are *InVircible's* own files. This floppy can be used to great effect when a virus has affected a hard disk.

Most of the product's features are accessible from within a single menu program. This has a cluttered interface, which continually displays a list of the amount of DOS memory available, the current integrity database filename, space available on the target disk drive, frequency with which integrity checks are made, and 'authorization status': i.e. whether or not copy-protected features have been installed.

Scanning

Buzzwords abound in the scanner as elsewhere in the product: for instance, the help feature for the scanner says that it is 'equipped with the SeeThru(c) anti spoofing feature and a generic boot code analyzer' - plain English would be infinitely more useful than this jargon.

When *InVircible* starts a scan, it displays the directory tree of the selected drive, then waits for the user to select a directory. If the root directory is chosen, the entire disk is scanned. Scanning an entire disk or a specific subdirectory seem to be the only available scanning options, and it is not possible to scan down part of a directory tree recursively.

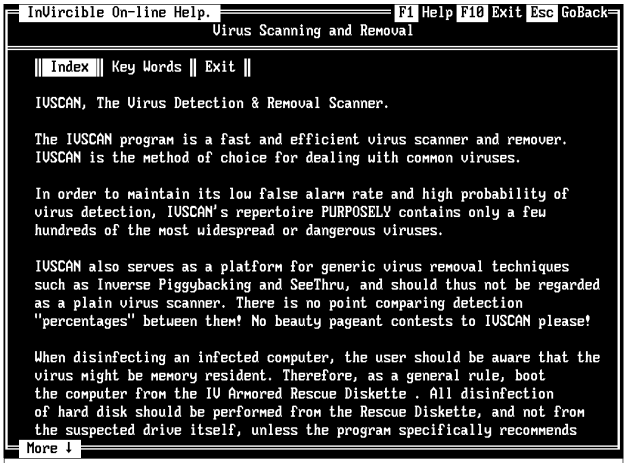
InVircible took 1 minute 15 seconds to scan the hard disk of my test PC, a timing which rose to 1 minute 19 seconds when the provided PIF file was used to execute the scanner in a DOS box under *Windows*. In comparison, *Dr Solomon's AVTK* took 1 minute 10 seconds to carry out the same task; *Sophos' Sweep* took 2 minutes 4 seconds in 'Quick' mode, 6 minutes 26 seconds in 'Full' mode.

Accuracy

The product detected 114 of the 248 infected test samples in the *Technical Details* - a mere 46%. Of viruses new to the last three test-set upgrades, six samples (21%) were detected; of those new to the most recent upgrade, none. My policy is to store infected samples with non-executable extensions: when I renamed files to an executable extension, another six (Durban, 1575, four of Number of the Beast) were detected as infected - I could not ascertain why.

None of the 500 Mutation Engine-infected test samples were detected, and although all nine boot sector viruses were spotted, Monkey, Italian, Form and Spanish Telecom were detected only as 'generic' infections. The fact that Form, which is the most prevalent in-the-wild virus, was not specifically identified, hardly inspires confidence.

The onscreen help says that the scanner 'PURPOSELY contains only a few hundreds of the most widespread or dangerous viruses'. This, and the manual, further justifies the abysmal performance of the scanner by saying that it should not be tested against virus collections - sadly, viruses are unlikely to take much notice of this plea.



```
IVSCAN, The Virus Detection & Removal Scanner.

The IVSCAN program is a fast and efficient virus scanner and remover.
IVSCAN is the method of choice for dealing with common viruses.

In order to maintain its low false alarm rate and high probability of
virus detection, IVSCAN's repertoire PURPOSELY contains only a few
hundreds of the most widespread or dangerous viruses.

IVSCAN also serves as a platform for generic virus removal techniques
such as Inverse Piggybacking and SeeThru, and should thus not be regarded
as a plain virus scanner. There is no point comparing detection
"percentages" between them! No beauty pageant contests to IVSCAN please!

When disinfecting an infected computer, the user should be aware that the
virus might be memory resident. Therefore, as a general rule, boot
the computer from the IV Armored Rescue Diskette . All disinfection
of hard disk should be performed from the Rescue Diskette, and not from
the suspected drive itself, unless the program specifically recommends
```

According to *Invircible's* on-line help system, IVSCAN 'PURPOSELY contains only a few hundreds of the most widespread or dangerous viruses'.

Integrity Checking

The manual states that the integrity checking software 'takes a 66 byte snapshot (signature) of critical information from each executable file'. *InVircible* affirms that it is able from this to verify the integrity of each protected file and repair damage caused by viruses. However, no details are provided of exactly what this signature is.

Verification of the integrity of the same hard disk used for the scanner tests takes 41 seconds under *MS-DOS*, and 43 seconds under *Windows* - about two-thirds of the time taken to scan the disk. Integrity checking is without a doubt the best part of *InVircible*. It works quickly and efficiently; however, it concentrates only on the beginning and end of files, where viruses are likely to act - alteration of bytes from about 2000h upwards is not noticed.

My main complaint with integrity checking is that *InVircible* creates a data file in each directory: it should maintain these files in its own directory. The documentation claims that this is a positive feature as, if a single database file is corrupted by a virus, all integrity checks are lost.

Disinfection

The integrity checker and the scanner offer several methods for removing viruses from infected files. None of these are available unless the software has been executed from the original floppy disk, or the 'authorization key' installed from the original floppy. *InVircible* refuses to install the authorization key unless it can write back to the original floppy. This is a seriously poor idea: master disks should be inviolate, and software which insists on writing back to what is the only working copy should be treated with contempt.

The developers state that 'anti-viral programs that lack good recovery features are a waste of time and money'. I disagree. Always, but always, replace an infected file with a clean copy of the original. Trying to mop up after a virus is always a hit and miss affair, and with many viruses, impossible.

Even if I could be persuaded that disinfection is a good idea, *InVircible* does not use the technique efficiently. Take, for example, the 66-byte entry created by the integrity checker for each file: no matter what information is stored there, if a virus affects a greater number of bytes, re-creation of the original will be impossible. The developers will no doubt argue that viruses usually affect the start/end of a file: true, but if disinfection techniques like those used in this product became commonplace, virus authors would soon take account of that fact, and alter their methods accordingly.

The scanner offers several methods for restoring original uninfected files. These only work, however, if the scanner detects an infection: as observed above, this does not happen often! Of the 114 test files detected as infected, *InVircible* claimed to have removed the virus from 81, deleted 15 and asked that the integrity checker be used to remove 15 more.

Both COM and EXE versions of Jerusalem were marked 'restored': *InVircible* appeared to distinguish between this action and virus removal. When the Necropolis virus was active, the message 'Please refer to the Manual' appeared onscreen - the manual was no help. Curiously, one sample of AntiCAD caused *InVircible* to remove the virus, restore the file, and then remove the virus again. All by itself.

The integrity checker's disinfection routine too was unsuccessful: no other files were flagged as restored, and only six were marked as having a different signature (Monxla, Butterfly, and two each of Murphy and of Sibel Sheep). The six viruses missed by the scanner when not stored as COM or EXE files were also not found by the disinfector, even if they were COM or EXE files.

Next, I tried to use 'Inverse Piggybacking', one of *InVircible's* special disinfection methods. More jargon. Hands up all those not intimately involved in virus research who know what this means. This feature thought that all files it had advised should be dealt with by the integrity checker were still infected. The manual further advises a user to 'start with the virus loaded in memory'. The author's strategy is obviously to have the virus active, using it to help with disinfection. While this could have limited usefulness in some cases, *VB* and this reviewer, for obvious reasons, would never recommend such techniques.

Designer Features

The tests performed by *InVircible* during the boot sequence ran into trouble with my multi-choice boot selection process. I sometimes use 4DOS in place of COMMAND.COM: if successive reboots swap between the two, *InVircible* produces an error message stating 'The COMSPEC date has changed, this may indicate an infection'. The fact that a different command interpreter is in use has not been noticed; the product merely complains that the date is incorrect.

InVircible has no memory-resident program: in its place is a program designed to run protection from within batch files. These would have to be created for everything requiring protection. Such programs would then have to be launched from within a DOS box: many (e.g. *Windows*) cannot be executed in this manner. This type of protection would be of little or no use in such arenas.

Another component, the Hyper-Correlator, aims to detect new non-polymorphic viruses by comparing code at the start of each file. The manual spends much time explaining what this is not, but fails to define what it actually is. I suspect that many users would have no idea what to do with it.

Features entitled 'ResQdisk' are provided to back up or restore partitions, using rescue disk information where possible. These seemed satisfactory, but their use is not for the faint-hearted. They are powerful, and can restore a disk's partition sector from a virus-created mess, but can also spell disaster if incorrectly utilized: using a rescue disk created on another PC may have spectacular side-effects.

Conclusions

This is the second consecutive month I have reviewed a product with a scanner which is exceedingly poor at virus detection. That the scanner is the dominant feature of neither product is immaterial: if developers include a scanner, it should work well. There are many good scanners available - *InVircible's* developers would be well-advised to license one if they cannot keep theirs up to scratch. Such a scanner, in tandem with other well-thought-out and efficient components, could help make a useful and versatile product.

As to the features to remove viruses from infected files: no matter how well they work, infected files should be replaced with known clean originals. Disinfection, inherently, is gambling, and I treat all such features with equal disdain. Even with copy-protection removed, given my measured results, which were well below average, I would find it difficult to recommend *InVircible*. Indeed, whilst it is copy-protected, my conclusion is the same as last month. Avoid.

Technical Details

Product: *InVircible v5.07A.*

Vendor: *New Castle International Corp*, PO Box 267, Rye Beach, NH 03871, USA. Tel. +1 603 431 6170, Fax +1 603 431 6370.

Availability: Not specified. A hard disk is used to install, but execution from floppy is possible. Many features are unavailable unless execution takes place from the original floppy disk.

Serial number: None visible.

Price: US$99.00 for a single user license.

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

Viruses used for testing purposes: This suite of 158 unique viruses (according to the virus naming convention employed by *VB*), spread across 247 individual virus samples, is the current standard test-set. A specific test is also made against 500 viruses generated by the Mutation Engine (which are particularly difficult to detect with certainty). For details of the test-set, see *VB* 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. 01235 555139, International Tel. +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email virusbtn@vax.ox.ac.uk
CompuServe 100070,1340@compuserve.com

US subscriptions only:

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

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

No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein.

END NOTES AND NEWS

Trend Micro Devices has announced that *Twinhead International*, the largest notebook manufacturer in Taiwan, will include *Trend's Mobile Protect*, **a file transfer utility with anti-virus protection**, in a pilot project of 1500 laptops. The decision was made after a detailed survey showed that almost two-thirds of respondents required protection against viruses, and over half needed a file transfer utility. *Mobile Protect* is the only product to offer both components in one package.

A combination of new features to help users **configure, control, and protect** their systems has been launched by *Central Point*, of *CPAV* fame. The company, which recently merged with *Symantec*, announced that *More PC Tools* was designed in response for additional capabilities in *PC Tools*. The product can be used either in addition to, or instead of, *Central Point's* already well-established *PC Tools*.

A transparent 'sentinel' program has been developed by *Absolute Software*, which uses **virus-like technology** to track a computer after it has been stolen. A user can register his machine with the *CompuTrace TRS* tracking line: when it is stolen, the service can pinpoint its whereabouts the next time the machine 'reports in'. Tel. +1 604 730 8951, Fax +1 604 730 9581.

The telephone number given in November's *VB* for ***S&S International's* German Anti-Virus Workshops** was incorrect. Please contact them on Tel. +49 40 2519 540 should you have queries about any of their workshops in that country.

IBM has announced it is developing an *Automated Immune System*, an electronic version of its human counterpart, which claims to be able to detect a virus with no previous knowledge of it. Jeffrey Kephart, manager of virus technology at the *IBM T J Watson Research Center*, described the product as **modelled on the way the human immune system protects against viral infection**. Contact *IBM* for further details. Tel. +1 914 945 3000, Fax +1 914 945 2141.

Rumours that **hackers obtained private telephone numbers** of the British Royal family have been denied by *British Telecom.* David Orr, of the *BT* press office, clarified the position: 'There were no hackers. A person temporarily employed by us is alleged to have illegally obtained some telephone numbers, and we are investigating the incident.' No complaints have been made as yet to the police; however, the company is considering its legal position in the matter.

Joe Wells, one of the **world's leading virus experts**, has joined the team at *IBM's T J Watson Research Center*, it has been announced within the past month. Wells said: 'I hope to be able to contribute to the development of anti-virus technology that will last into the next century.' Wells, a well-known and liked member of the small anti-virus community, is best known for his maintenance of 'The wild list', a summary of those viruses known to be in the wild.

A story published on 10 November 1994 in *Computer Weekly*, which stated that the man alleged to be the **Black Baron (author of SMEG)** had skipped bail, has been refuted by the *Computer Crime Unit*. DS Simon Janes confirmed that the virus author did appear in court as scheduled, and was re-bailed until 10 January 1995, by which time the *CCU* hopes to have completed its investigations.

The German *BSI* is in the process of publishing its **definitions of computer terminology**. The latest release defines what constitutes a computer virus, listing and describing the different types, as well as terms in common use in the field, such as write-protection, CMOS RAM, and even CERT (taken over from the English).

The new Data Protection Registrar in the UK, Elizabeth France, has outlined some of her plans for **the future of the *Data Protection Act*.** Commenting on her appointment, France noted that 'The *1984 Data Protection Act* no longer easily fits the demands of current business practice and technology', and plans to streamline procedures.