January 1990 ISSN 0956-9979

VIRUS BULLETIN

THE AUTHORITATIVE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Edward Wilding**

Technical Editor: **Fridrik Skulason,** University of Iceland

CONTENTS

EDITORIAL

AIDS Information Version 2.0

On various dates between December 8th and 12th of last year several thousand 'AIDS Information' disks were posted from London to computer users throughout the UK, Europe, Africa, Scandinavia, Australia and possibly other countries. The software which contained an 'interactive' questionnaire about the AIDS/HIC biological virus also contained a pernicious Trojan horse program designed to encrypt sections of the root directory of a PC's hard disk

In the United Kingdom these disks had been posted to the subscriber list of *PC Business World* magazine. Needless to say, the list had been obtained by deception. Credit must go to management at *PC Business World* who immediately contacted Jim Bates, a member of the *Institution of Analysts & Programmers*, and asked him to undertake a disassembly of the disk. Jim, a regular contributor and editorial advisor to the *Virus Bulletin*, proceeded to conduct a most thorough and painstaking analysis of the AIDS Information disk. In the course of his investigations he produced both a reliable removal program called AIDSOUT and a retrieval program (CLEARAID) to recover encrypted material after the Trojan horse has been triggered. Due to his efforts, and those of his colleagues and friends who provided support and assistance, the damage wrought by these disks has been minimised. Analysing such a large program is a far more intensive and time consuming process than disassembling a computer virus. Viral code can be measured in hundreds of bytes but a Trojan element or logic bomb(s) can reside anywhere within thousands of bytes of code. This analysis was an immense undertaking and we should like to thank Jim for the many hundreds of hours he spent analysing the AIDS disks and for producing reliable programs with such speed and efficiency.

The AIDS disks appear to have been part of an elaborate if rather clumsy attempt to extort money from unsuspecting PC users. The extent of this extortion attempt is unparalleled with disk mailed worldwide. Delegates to a world congress on the AIDS biological virus held in Stockholm in October 1988 had been targeted prompting the *World Health Organisation* to put out an alert. Subscribers to French and Italian business magazines had also been mailed. There has been much speculation as to the identities of the perpetrators. Regrettably, there have also been a number of totally inaccurate, unjustified and false accusations levelled against entirely innocent organisations and individuals.

Some salutory lessons have been learned. Unsolicited software will, in the future, be treated with extreme caution by many. It is extraordinary that controls to prevent use of these rogue disks failed, or never existed, in so many businesses. It is also worth pointing out that virus specific software is useless for tackling unknown threats. Only checksumming programs revealed the corruption of the AUTOEXEC.BAT caused by the AIDS disk. Finally, companies should take note that they are vulnerable to the misuse of their mailing lists and a reassessment of list management by many companies and list brokers may be warranted.

It was, perhaps, inevitable that the profiteers would exploit the situation. Some individuals and companies were quick to offer "AIDS clearing" and "AIDS protection" services - at a price - to affected users. Many of these offers were made based on the flimsiest knowledge and sometimes without any analysis whatsoever! Conversely, the national UK newspapers showed welcome restraint in reporting this incident. Fortunately the tabloids seem to have 'missed the boat' on this one. Anyone who saw the *Daily Star's* Britain on the Blink' Datacrime exclusive on October 1989, will heave a sign of relief.

New Scotland Yard's *Computer Crime Unit* are heading the international investigation. The unit is anxious that people who received the disk should contact its officers and send them the packaging and contents - preferably untouched - for forensic examination. Detective Inspector John Austen is heading the enquiry and can be contacted on telephone number 01 725 24 09 (international + 44 1 725 24 09)

A highly condensed technical analysis of the AIDS disk appears in this month's edition of the *Virus Bulletin*. A full dissection of the disk is contained in a 40 page report compiled by Jim Bates. The report is free of charge and can be obtained by faxing or writing to the *Virus Bulletin* (address on back page). Please mark correspondence 'JIM'S REPORT'. We regret that requests for the report cannot be taken by telephone.

The AIDSOUT disk is available in over 90 countries worldwide. It is supplied free of charge. In the United Kingdom it is available from Robert Walczy, AIDSOUT office, CW Communications, 99 Grays Inn Road, London WC1X 8UT. Tel 01 831 9252. Fax 01 405 2347. BBS 01 831 6221. Overseas subscribers are advised to contact their local *IDG Communications* office.

TECHNICAL ANALYSIS

This article provides a concise analysis of the AIDS Information disk from PC Cyborg Corporation which contains a Trojan horse.

Jim Bates' full report about this disk is available free of charge from Virus Bulletin. Please refer to Editorial on page 2.

Trojan Horse: AIDS Information Introductory Diskette Version 2.0

Jim Bates

The AIDS Information disk contained two program files, INSTALL.EXE and AIDS.EXE which were both written in QuickBASIC 3.0 and compiles with the supplied compiler (v5.60) standard manner. In spite of intensive investigation, no malicious code has been found in the AIDS.EXE program, its only claim to fame being that it requires the INSTALL program to be run before it can operate. The INSTALL.EXE program on the other hand was compiled from at least seven different source modules and performs at least five different functions. These can be described as **Installation, Counting, Sharing, Triggering** and **Faking**. Before examining each individual phase, it should be mentioned that the program only appears to affect the C: drive on the machine, although during **Installation** and **Sharing**, write access is required to the floppy disk in order to mark the program's progress.

Installation

Taking each phase in turn, when the program is first installed from the distribution disk it creates a series of hidden directories on the C: drive, variously named using a combination of spaces and ASCII character number 255 (OFF hex). It should be noted that ASCII character 255 displays on most systems as a space. There are five of these directories, all descended from the root directory of the C: drive and they are laid out as follows:

1 - C:\#
2 - C:\###s###
3 - C:\###s###\##s####
4 - C:\###s###\##s####\####s##
5 - C:\###s###\##s####\####s##\ERROR IN THE

The numbers are for reference only and note that I have used the hash character (#) to indicate the ASCII 255 character and lower case 's' to indicate spaces.

Once these have been created, the program then continues by creating a number of files in the directory referred to as number 4 above. The filenames are composed of combinations of spaces and underscore characters and are listed here with the '@' character representing underscore and lower case 's' representing spaces. The apparent function of each file is also noted.

1 - @.ss@ Count of "application" leased by the user
2 - @.s@ Reboot counter file
3 - @.s@@ Used to reserve disk space for later operations
4 - @@@.ss@ Contains nothing - probably a flag file
5 - @@@@.s@@ Contain program "serial numbers"

Once again the numbers are for reference only.

File 3 is 50401 bytes long, containing spaces, and is written to reserve space which will be required by the program during the trigger phase. If there is insufficient space for all these control files, the installation program exits with the following message:

```
Sorry, the had disk drive C is too full.
Remove a few old files from it and then try again.
```

File 5 is necessary to ensure that a specific renewal disk can only be effective on one particular installation. The serial numbers are generated from the system time and date

functions and are noted in the invoice which the installation phase prints.

The installation then continues by making a copy of itself under the name REM#.EXE in the directory number 1 above (note that once again the hash character indicates ASCII character 255). It should be noted that when examined using most utility programs, the ASCII 255 character appears as a space.

The AIDS.EXE program is then copies from the floppy disk into the root directory of the C: drive.

The final stage of the installation then goes on to manipulate the AUTOEXE.BAT file in the root directory of the C: drive in the following way. The existing AUTOEXEC.BAT is rewritten as AUTO.BAT and three lines of remarks are added at the beginning of the file. Then a file named AUTOEXEC.BAK is created (and hidden) and this contains the text "File not found". Finally, a new

AUTOEXEC.BAT file is created (and hidden) and this contain the following text:

```
echo off
C:
cd\#
rem# PLEASE USE THE auto.bat FILE INSTEAD OF
     autoexec.bat FOR CONVENIENCE
auto.bat
```

Note the ASCII character 255 (represented by the hash sign) on the third and fourth lines.

Having completed the file setup on the C: drive, the program then goes on to print an invoice for payment to made for "leasing" this software to t PO Box number in Panama.

This completes the Installation phase.

Counting

Once installed, the program apparently has no effect on the operation of the machine. However, each time the new AUTOEXEC.BAT file is executed (ie: during each machine reboot) processing is routed through the REM# program as a result of the two lines which contain the ASCII character 255. The first of these (cd\#) changes the default directory to the one containing REM#.EXE file, then the line containing the REM# command is executed. Normally, the command REM (short for REMARK) is a signal to the batch processing facility of the command processor that any text following is to be ignored. However, in this instance, while the REM# *appears* to be an ordinary REM command, the addition of the ASCII character 255 means that the command processor does not recognise it as such and therefore attempts to load and execute a program of which matches the augmented name. Since a program exists which is called REM#EXE, no error occurs and the program runs quite legally. When run, the program recognises from examination of the files on the disk that the installation phase has been completed but the trigger phase has not yet been reached. Under these circumstances the program's only function is to increment a counter in one of its control files and report when that counter reaches maximum. In tests, the usual number of reboots necessary to reach maximum has been 90 but slight variations on this, coupled with reports of lower numbers being seen indicates that a random element may be used when the counter start number is first generated (during installation).

Until the counter reaches maximum, the REM# program simply exits and processing continues with the remaining commands listed in the AUTO.BAT file (which was the original AUTOEXEC.BAT).

Triggering

Once the program senses that the counter has reached maximum, it enters the trigger phase. The purpose of this phase is to prevent further use of the machine until a "renewal" disk can be provided after payment for "leasing" the software is made. This is achieved under the control of a module of the REM# program as follows:

Several more control files, using similar underscore/space naming conventions to those generated during the installation phase, are written to the disk. These are variously listed as follows:

T1 - @@.s@ Contains DOS version message
T2 - @@.s@@ Contains 5 records of 250 spaces each
T3 - @@.ss@ Generated listing of all filenames (see below)
T4 - @@@.s@ Contains text "IO.SYS"
T5 - @@@@.ss@ Contains listing of all directory names
T6 - @@@@.s@ Packing file

These files are numbered with a prefix 'T' to distinguish them from the control files generated during installation.

Prior to these files being written, the program deletes the file referred to as file 3 (space reserving) created during the installation phase. This will ensure that at least 50K of space is available on the disk for the subsequent operations.

Communication with the DOS interface within QB3 was awkward and no specific commands were available to collect file and directory specifications from the disk, other than a simple FILES command which only listed file names to the screen. However, most QB3 programmers were aware of the SHELL command which enables direct communications with DOS, albeit under the auspices of a child command processor. The contents of files T1, T3 and T5 were apparently generated by a series of redirected SHELL commands which placed their output into the relative files where they could be accessed by the program later.

File T6 has a variable length and is generated by the program to pack the disk to the point where there is no more free space available. This actually contains records consisting mostly of spaces but where each record has the header ‘DATA RECORD #’ which is followed by a number. The numbers follow an apparently random sequence and were probably generated by a timing loop designed to ensure that the packing process took a similar length of time regardless of the speed of the machine processor.

Once these control files have been written, the program moves on to complete the locking up of the machine by examining each filename on the disk in turn and encrypting its name according to the following criteria:

The extension portion of the filename is checked to see if it exists within a table maintained by the program. If it does, the extension is changed to a matching entry in a parallel table of extension names. This parallel table is characterised by each entry consisting of three characters, the first of which is a space and the remaining two are alphabetic characters. Once the extension has been changed, the filename portion is then encrypted on a character by character basis according to another pair of tables maintained by the program. If the extension is **not** recognised, then encryption does not take place on the filename. Each filename is then reset within the directory with its attribute bytes set to Read Only and Hidden. As this process continues, directory names are also set to Hidden although they are not encrypted in any way.

The system files and COMMARND.COM are specifically excluded from the encryption process since if they were encrypted, the machine would be unable to boot.

This completes the Trigger phase.

Faking

Once the trigger phase has completed, the program moves directly into the Faking phase. This is achieved under the control of yet another module within the program and its purpose is to fool the user into thinking that he is within the DOS environment while the program actually has control. If the machine is rebooted by switching off and then on again, it appears to boot directly into DOS but with a repeating message warning that "The lease for a key software package has expired" and further action could result in the destruction of "all of the files on drive C." What is actually happening is that the program (still the REM#.EXE program invoked by the hidden AUTOEXEC.BAT file), having recognised that the trigger has completed, is duplication the DOS environment response and reacting to many commands in a similar way while intercepting and preventing many others. For example, if the command DIR is entered, the user is shown what appears to be a correct file listing of the root directory of his C: drive. However, within this listing is a file called READ.ME which does not actually exist on the disk. If the user is fooled and enters the command TYPE READ.ME in an attempt to examine the contents of this file, he will see:

```
You are advised to stop using this computer. The software lease
has expired. Important: Renew the software lease before you use
this computer again.
```

This is just one example of the Faking phase, there are several others.

If the user attempts a reboot, using the Ctrl-Alt-Del key combination, the program traps this and fakes a reboot before returning to its previous deceptions. The only way out of this is to switch the machine off and then reboot from a system floppy disk. If this is done and the C: drive is examined, only a single file called CYBORG.DOC appears to be present on the disk and there is no free space available. This file is a plain text file and contains the familiar exhortation to "Renew the software lease".

However, closer inspection with special utilities reveals that all of the files and directories are still present although they all have their hidden attributes set and mot of them have been encrypted. The program does hot alter the contents of any of the user’s files, just their names.

It follows therefore, that restoration can be quite simply achieved once the extension and filename encryption tables are known. The process could be a long one and would require the use of special utilities to change the file attributes and make them available again.

"While the conception is ingenious and extremely devious, the actual programming is quite untidy"

The Program File

During my examination of the program under both static and dynamic conditions, several points have arisen which are worth mentioning.

While the conception is ingenious and extremely devious, the actual programming is quite untidy. This seems to indicate that the programmer was unfamiliar with the capabilities of QuickBASIC 3.0 and not particularly well schooled in general program construction. I suspect that the AIDS.EXE and INSTALL.EXE programs were written by different programmes as the style and use of program facilities differs quite markedly.

The INSTALL.EXE file contains a modified version of the USERLIB interrupt handling interface which is supplied with QB3. I have identified three references within the code to this powerful routine, two of which allow examination and modification of file attributed, while the third enables the program to poll the status of a printer attached to the LPT1: device. None of these facilities are available within the standard core of QuickBASIC 3.0

Normally, the display strings of characters used within a QuickBASIC program can be seen quite plainly in the data section of the compiled program file. When examining such files, a list of these strings is a valuable aid in determining what the program does under different conditions. However within INSTALL.EXE, the strings have been encrypted in such a way that casual examination reveals nothing intelligible. The encryption algorithm is a fairly simple one and once it has been solved, the strings can be displayed in the normal way. This immediately throws light upon some of the less obvious functions in the program, particularly concerning which commands are intercepted by the fake DOS module.

Surprise has been expressed in some quarters that a BASIC program is capable of intercepting the reboot key combination and it has been erroneously suggested that this program "cleverly" installed its own interrupt handler for keyboard access. The true fact is that the only interrupt re-vectoring which occurs within this program is that which is installed quite normally by QB3 (a total of 48 Get/Set vector calls in all). The trapping of the reboot combination is quite a simple process and can be achieved in a single line of BASIC code which is listed in at least one of the Microsoft reference manuals. The interesting point however, is that such trapping entails the use of a particular option within the compiler which enables "event trapping" on a statement by statement basis and this option is immediately obvious when the program file is disassembled to assembler level.

It has also been quite seriously suggested that since the program contains the names of the days of the week, that it could contain a virus triggered to activate on a day/date basis. This is also untrue, the days of the week are there to enable the faking module to respond correctly to the DATE command.

Some researches appear to favour a theory that this program contains an unidentified virus which may be deposited in the system during one of the phases of its activity. To counter such irresponsible statements and after hundreds of hours work investigating this program, I am prepared to go on record as saying that in my opinion there is definitely no virus present in the code. This means that once the control files, directories, encryption etc. have been removed, no trace of the program's activity will remains to replicate and cause further problems.

Jim Bates - January 1, 1990

Remedy programs

Two programs, written by Jim Bates are available free of charge.

AIDSOUT is a program to remove the AIDS Trojan horse program thus restoring an affected PC system to normal.

AIDSCLEAR is a decryption routine to recover scrambled files in the event that the trigger mechanism has activated.

The disks are available in the UK from: Robert Walczy, AIDSOUT Officer, CW Communications, 99 Grays Inn Road, London WC1X 8UT. Tel 01 831 9252, Fax 01 405 2347. The AIDSOUT disk is also available from IDG officers worldwide. IDG is the parent company of CW Communications.

SPECIAL FEATURE

Dr. Keith Jackson

Electronic Computer Conferencing and the AIDS disk

While the AIDS disk was being received, there was a pressing need for immediate information about its effect(s). Many TV and radio news bulletins, and daily newspapers, carried articles about the disk. In the main journalists restrained themselves from writing lurid nonsense. However, as the disk was new, much of the information was either speculative, later shown to be only half the story, or simply untrue. Only serious discussion can remedy this lack of knowledge, and this requires a flow of information. In retrospect, the UK system known as CIX (**C**ompulink **I**nformation e**X**change, telephone 01399 5252, any modem speed) acted as the main focus in the UK for discussing the AIDS disk.

CIX is an electronic conferencing system which contains an electronic mail system, and various discussion areas (conferences) related to specific topics. One of the most active conferences is devoted to computer viruses, and this proved to be a natural home for reporting the effects of a Trojan horse such as the AIDS disk. CIX has about 3000 registered users, most of whom seem to come from a technical background, with a preponderance of people who work in, or write about, the computer industry.

The first message describing the AIDS disk as a Trojan horse was placed on CIX on 12th December 1989 at 8:33pm. Half an hour later, another message confirmed that a second CIX user had received on of these disks, and within two hours a computer journalist was asking for further information and copies of the disk. Things exploded from there. Up to Thursday 21st December 1989, 376 separate messages had been placed on CIX about the AIDS disk (over 40 messages per day).

Although I did not receive an AIDS disk, I learned via CIX if its existence before many of the disks had arrived at their intended destination. I also know that the disk arriving at most UK addresses were mailed in South Kensington, sent with 20 pence stamps on Monday 11th December and contained only a ‘licence agreement’ and a floppy disk, which should not be used under any circumstances. Estimates of the total number of disks involved levelled out at about 10,000. (*informed estimates have since risen to nearer 20,000., Ed.*) someone had sat down and licked a vast number of stamps - a means to avoid being traced since UK franking machines are registered by the Post Office. It rapidly became clear that a mailing list was involved, and from discussion of which magazines were read by the various CIX users who had received the AIDS disk, *PC Business World* was the prime candidate. This was later confirmed by *PC Business World.*

Some CIX users are solicitors, and they participated in what proved to be a most interesting discussion. When the AIDS disk is run, it prints an invoice and asks you to send money to a box number in Panama as payment for using the AIDS questionnaire program. The penalty for not doing this is that your hard disk is encrypted (after a certain number of reboots), thus making all the files inaccessible. Solicitors emphasised that given sufficient warning about the possible consequences, and as long as the user can choose not to use the software, such a demand for payment may be legal. The ‘licence’ accompanying the AIDS disk states very clearly that it will:

"use program mechanisms to ensure termination of your use of the programs. These program mechanisms will adversely affect other program, applications……..your microcomputer will stop functioning normally."

Whether or not this makes such a scheme legal could only be tested in a court of law. One person even quoted what he thought were the relevant sections of The Criminal Damage Act 1971.

CIX messages contained a details technical description of how the AIDS disk works. This was culled from many contributors, and discussed in depth. Jim Bates (who writes elsewhere in this issue of *Virus Bulletin*) was the first person to write a program capable of removing the AIDS disk. This was distributed free of charge via his own Bulletin Board system, (*The Power Tower*., Tel 0533 880114), and a copy was also uploaded to CIX.

One rather apt and witty message simply declared "Timeo Danaos etdona ferentis" ("Beware of Greeks bearing gifts").

In conclusion, computer conferencing has evolved into a useful way of discussing complex subjects in a timely manner. It can wander off into trivia, produce false reports, and be downright misleading (as can all human conversation), but it has the major advantage that you can talk at first hand to the people at the leading edge of the problem, be they software authors, journalists or engineers.

LEGAL ISSUES

A Computer Crime Bill is likely to be introduced during this session of the UK Parliament by Michael Colvin MP. Legislation against computer virus writers and distributors has already been implemented in some States in the USA. Hacking has been made a criminal offence in many European countries and a recent English Law Commission report has recommended that English law follows suit and criminalises unauthorised access to computer systems. The AIDS Disk, now understood to be a case of computer blackmail without parallel has added impetus to implementing legislation. In this article Owen Keane, a barrister specialising in computer law, discusses the legal issues surrounding the AIDS Information disk. The author has asked us to point out that material in this article should not be treated as formal advice. The PC Cyborg Corporation is referred to throughout as Cyborg.

AIDS Disk - Issues Pertaining to English Law

Owen Keane

The AIDS disk has heightened awareness of the need for new statutory measures to protect the special circumstances of computer use. This appears to be the first attempt at commercial exploitation of Trojan horse programming. In the absence of specific new measures we must consider the present position. There are two aspects to liability; **criminal** and **civil**.

Criminal liability is initially a matter for police investigation, here by the Computer Crime Unit within the City and Metropolitan Fraud Squad reporting to the Director of Public Prosecutions. Their report will recommend prosecuting of offences disclosed for which there is sufficient evidence. Offences considered may include **criminal damage, obtaining by deception** or **fraud**, and possibly cheating or blackmail. Evidence is the police's responsibility, including program investigation which may require expert assistance. A private prosecution may be launched by application for a summons supported with evidence.

All offenders have two elements, the physical act and the state of mind of its performer. Here we are immediately confronted with difficulties; the acts complained about are done by the computers installed with the AIDS programs. Further the complainants or their staff installed them, not the sender. Thus there is an immediate break in the chain of causation between the sender and the matters complained of.

Previously, prosecutions have been brought for criminal damage to computer programs. Some of these prosecutions have succeeded. In *Cox v Riley*, a 1986 case, the erasure of a computer program from a plastic circuit card was considered criminal damage; the wood-saw it controlled being rendered inoperable. In the present case the encryption is by the computer, at the triggering of what seems a fairly simple but effective "logic bomb". To the best of my knowledge there has yet to be a successful prosecution in the UK for criminal damage by means of a logic bomb. There are differences between the classic logic bomb situation and the instant facts. Normally the person who inserts the extra code is its author. Usually, there is no warning about adverse effects which may result. The sender of the AIDS disks has warned of its effects in at least three places, albeit misleading.

The 'documentation' which accompanies the disk states that it may adversely affect the other programs on the computer once installed. The point here is that the time scale on which adverse effects occur is unrelated to how often the Aids programs are used.

There seems little doubt that sections of the hard disk are deliberately "scrambled" or encrypted. There are two simple statements warning of adverse effects and halting of normal functioning. To prove when the author of the programs intended to do this will require precise technical evidence detailing the inter-relation and operation of the counter and encrypting routines.

Crucial to proving the intention of recklessness necessary for criminal damage will be evidence of the harm's timing. Another consideration is the creation and contents of seemingly unnecessary subdirectories, concealed from the user. The surreptitious counter mechanism therein, which seems to have no link to the number of times AIDS.EXE is run, is of key importance. If it only prevented use of AIDS.EXE after the appropriate number of runs (Cyborg's term "user applications" is unsatisfactory) it may be a legitimate means of protecting distributor revenue. This does not appear to be the case. Its sole intent - counting bootstraps performed and not runs of AIDS-EXE - has no sustainable reasoning as in any event the number is not consistent and not 365.

Furthermore there appears to be no means of instructing the program which form of licence recipients intend to take up. This casts further doubt on the purpose of any counter mechanism.

One weapon the prosecution would rely on heavily is the principle that a person intends the natural consequences of his acts. Here it could be used to good effect as it seems inevitable that after the recipient has received the disk and obeyed the instructions his hard disk will be trashed after a random number of boots. However, that will no doubt be countered by the defence is saying that they didn’t force the person to use the disk or do it themselves; giving someone a gun is not the same as shooting them.

There are similar problems with blackmail; under the statutory definition there has to be an unwarranted demand with menaces. Here the supplier has a right to the licence fee, so the veiled threats in the accompanying literature are non-effective. The definition of menaces is wide enough to include threats to property, even the intangible.

Turning to obtaining by deception there are also hurdles to be overcome. The statutory definition, in the 1968 Theft Act requires the deception to be on a human and to precede the obtaining. Furthermore, it must be proved that the false statement, or conduct, caused the victim to have acted in the appropriate way. There are also limitations on how remote the obtaining is from the deception. When someone has obtained a position by deception they are not guilty of obtaining their salary by deception, it is received for services rendered instead. The situation is covered by obtaining a pecuniary advantage by deception. However the definition of pecuniary advantage does not cover this situation. Where is the deceit? There are two possibilities; the representation that the minimum period for which the program will work is "365 user applications" and the implied representation that the adverse effects on other programs will not take place unless the user is in breach of the terms of the "lease". Another issue will be what has been obtained. Unless money has been sent in consequence of the representations no prosecution for the substantive offence will succeed but a charge for attempt may lie.

On a more practical level there may be difficulty in finding and proving who posted the disks and/or who wrote the code, or was responsible for its inclusion or knew its effects. If it can be proved that the distributor (or programmer, if they be separate) knew of the effects of installation then he may be liable for prosecution.

If Cyborg is not a registered company then its members can be charged with any offence elected to be proceeded with, either collectively, or individually if it can be shown that any individual did sufficient to amount to an offence on his own account. Any charge of conspiracy will require the prosecution to prove an agreement to a course of conduct which necessarily involved the commission of an offence, and to prove such agreement beyond reasonable doubt. This is a heavy burden.

If any prosecution were to succeed, any person whose computer was damaged could apply for a compensation order if their computer was the subject matter of a charge or an offence taken into consideration. There may be limitations and argument as to compensation’s availability as it is normally used in cases of physical of direct financial loss such as theft. Further still the court must have regard to the defendant’s means.

Civil liability has possibilities, but as with criminal liability some difficulties and limitations may prevent satisfactory redress. Possible causes of action lie in **tort** and **breach of contract.** Immediate advantages of civil actions include only having to prove facts alleged to be true on the balance of probabilities, and the recoverability of damages.

An important point to bear in mind is that neither the disk nor the programs on it are sold. Instead a licence is offered. The seemingly standard literature accompanying the disk purports to limit Cyborg’s liability to replacing the disk. It is not possible to limit liability to this extent. Excluding liability for death or personal injury is never allowed. Cyborg’s attempt too pass all risk for the use of the programs, including liability for ant type of damages to the user is wholly invalid. By virtue of the Unfair Contract Terms Act, limitation of Cyborg’s liability will be controlled by the reasonableness test. Previous case law has rendered the results of this test unpredictable. Reasonableness is decided in the light of statutorily defined criteria, most of these will favour recipients.

An action for breach of contract may succeed for less than full performance of the 365 minimum uses under the terms of the licence, or for breach of the implied warranties that they will conform to their description and be fit for their purpose.

However, in the present circumstances if it is clear that the premature failure of the software was deliberate and that the whole purpose of the scheme was a deliberate sham the Courts are unlikely to treat Cyborg’s terms with great sympathy. The most unreasonable terms are the total exclusion of liability for mis/non/partial performance, and the mere notice of unspecified damage to the disk’s other contents in the event of breach of the licence. Cyborg effectively reserve the right to prevent use of any programs or data on the machines until the licence is renewed. There is a difference between renewing the licence to use a program on exposure to AIDS and being forced to pay to regain control of independent programs and data.

In additional an action may lie in tort. Under an action grounded in fraudulent misrepresentation or intentional damage Cyborg may be liable for compensatory damages. Such damages would cover the cost of restoring or replacement of all the data and or programs on any affected machine, if the machine were to be damaged then this would be covered as well. The underlying aim being to put the recipient in the position as if no misrepresentation or damage had happened.

PC Cyborg's 'licence agreement' (enlarged!)

Limited Warranty

If the diskette containing the programs is defective, PC Cyborg Corporation will replace it at no charge. This remedy is your sole remedy. These programs and documentation are provided "as is" without warranty of any kind, either express or implied, including but no t limited to the implied warranties of mechantability and fitness for a particular purpose. The entire risks as to the quality and performance of the programs is with you, Should the programs prove defective, you (and not PC Cyborg Corporation or its dealers) assume the entire cost of all necessary servicing, repair or correction. In no event will PC Cyborg Corporation be liable to you for any damages, including any loss of profits, loss of savings, business interruption, loss of business information or other incidental, consequential, or special damages arising out of the use of or inability to use these programs, even if PC Cyborg Corporation has been advised of the possibility of such damages, or for any claim by any other party.

License Agreement

Read this license agreement carefully. If you do not agree with the terms and conditions stated below, do not use this software, and do not break the seal (if any) on the software diskette. PC Cyborg Corporation retains the title and ownership of these programs and documentation but grants a license to you under the following conditions: You may use the programs on microcomputers, and you may copy the programs for archival purposes and for purposes specified in the programs themselves. However, you may not decompile, disassemble, or reverse-engineer these programs or modify them in any way without consent from PC Cyborg Corporation. These programs are provided for your use as described above on a leased basis to you; they are not sold. You may choose one of the types of leases (a) a lease for 365 user applications or (b) a lease for the lifetime of your hard disk drive or 60 years, whichever is the lesser. PC Cyborg Corporation may include mechanisms in the programs to limit or inhibit copying and to ensure that you abide by the terms of the license agreement and to the terms of the lease duration. There is a mandatory leasing fee for the use of these programs they are not provided to you free of charge. The price for "lease a" and "lease b" mentioned above are US$189 and US$378, respectively (subject to change without notice). If you install these programs on a microcomputer (by the install program or by the share program option or by any other means), then under the terms of this license you thereby agree to pay PC Cyborg Corporation in full for the cost of leasing these programs. In the case of your breach of this license agreement, PC Cyborg Corporation reserves the right to take legal action necessary to recover any outstanding debts payable to PC Cyborg Corporation and to use program mechanisms to ensure termination of your use of the programs. These program mechanisms will adversely affect other program applications on microcomputers. You are hereby advised of the most serious consequences of your failure to abide by the terms of this license agreement: your conscience may haunt you for the rest of you life; you will owe compensation and possible damages to PC Cyborg Corporation; and your microcomputer will stop functioning normally. Warning: Do not use these programs unless you are prepared to pay for them. You are strictly prohibited from sharing these programs with others, unless: the programs are accompanied by all program documentation including this license agreement; you fully inform the recipient of the terms of this agreement: and the recipient the recipient assents to the terms of the agreement, including the mandatory payments to PC Cyborg Corporation. PC Cyborg Corporation, then do not use these programs. No modification to this agreement shall be binding unless specifically agreed upon in writing by PC Cyborg Corporation.

Programs © copyright PC Cyborg Corporation, 1989
Compiler runtime module © copyright Microsoft Corporation, 1982, 1987
All Rights Reserved.
IBM© is a registered trademark of International Business Machines Corporation. PC/XTtm is a tradmark of International Business Machined Corporation. Microsoft and MS-DOS© are registered trademarks of Microsoft Corporation.

When considering whether to sue (are they worth suing?), and under what grounds, remember the limitation on recoverable damages. Contractual damages are limited to putting the recipients in the position they would have been in if the licence agreement had been fulfilled. The premature termination of the ability to assess exposure to AIDS may not be of great concern or value, but the loss of access to other information may well be. Contractual damages will only cover the former. Recovery for the latter will be by tortious damages, these again are limited, preventing recovery of pure economic loss. Economic loss is any loss not directly resultant from physical damage. In the context of machines, it prevents the recovery of lost profit from time not in operation, apart from the initial interference with a task in progress. It will however cover the re-installation of data and programs to restore the computer to the position it was before receipt of the AIDS disk.

As the disk represents unsolicited goods the duties of the recipients are those of an involuntary bailee - there is not duty to return the disk, you must not prevent its repossession by Cyborg. As it's not your property you cannot destroy or dispose of it but you have no duty to protect it. If recipients give Cyborg notice that they do not want the disk or Aids programs, and Cyborg do not take back the disk, it will become the recipient's property after 30 days if not used for business.

Lastly, copyright in the programs will remain with Cyborg.

IBM PC VIRUS PATTERNS

The following are hexadecimal patterns of known viruses affecting IBM PCs and compatibles. This can be used to detect the presence of the virus by the "search" routine of disk utility programs such as The Norton Utilities or your favourite disk scanning program (See *VB Nov 89*).

Seen viruses

405	26A2 4902 26A2 4B02 26A2 8B02 504B 19CD ; Offset 00A
Alabama	8CDD 33DB 8EDB 8B07 0B47 0274 7489 1F89 ; Offset 109
Brain	A006 7CA2 097C 8B0E 077C 89OE 0A7C E857 ; Offset 158
Cascade (1) 01	0F8D B74D 01BC 8206 3134 3124 464C 75F8 ; Offset 113, 1701 bytes, Falling characters
Cascade (1) 04	0F8D B74D 01BC 8506 3134 3124 464C 75F8 ; Offset 113, 1704 bytes, Falling characters
Cascade (1) Y4	FA8B CDE8 0000 5B81 EB31 012E F687 2A01 ; Offset 100, 1704 bytes, Falling characters
Cascade format	0F8D B74D 01BC 8506 3134 3124 464C 77F8 ; Offset 113, 1704 bytes, Formats hard disk
Dark Avenger	740E FA8B E681 C408 08FB ; Offset 068, 1800 bytes
Datacrime (1)	3601 0183 EE03 8BC6 3D00 0075 03E9 0201 ; Offset 002, 1168 bytes
Datacrime(2)	3601 0183 EE03 8BC6 3D00 0075 03E9 FE00 ; Offset 002, 1280 bytes
Datacrime II	2E8A 072E C605 2232 C2D0 ; Offset 022, 1514 bytes
dBASE	50B8 0AFB CD21 3DFB 0A74 02EB 8A56 E800 ; Offset 636, 1864 bytes
dBASE destroy	B900 01BA 0000 8EDA 33DB 50CD 2658 403C ; Offset 735, 1864 bytes
Den Zuk	FA8C C88E D88E DOBC 00F0 FBB8 787C 50C3 ; Offset 0
Disk Killer	2EA1 1304 2D08 002E A313 04B1 06D3 E08E ; Offset 0C3
Fu Manchu	FCB4 E1CD 2180 FCE1 7316 80FC 0472 11B4 ; Offset 1EE, 2086 bytes COM, 2080 bytes EXE
Icelandic (1)	2EC6 0687 020A 9050 5351 5256 1E8B DA43 ; Offset 0C6, 656 bytes
Icelandic (2)	2EC6 0679 0202 9050 5351 5256 1E8B DA43 ; Offset 0B8, 642 bytes
Icelandic (3)	2EC6 066F 020A 9050 5351 ; Offset 106, 632 bytes
Italian-Gen	B106 D3E0 2DC0 078E COBE 007C 8BFE B900 ; Offset 030
Italian	32E4 CD1A F6C6 7F75 OAF6 C2F0 7505 52E8 ; Offset 0F0
Jerusalem	03F7 2E8B 8D11 00CD 218C C805 1000 8ED0 ; Offset 0AC, 1813 bytes COM, 1808 bytes EXE
Lehigh	8B54 FC8B 44FE 8ED8 B844 25CD 2106 1F33 ; Offset 1EF
Mistake	32E4 CD1A 80FE 0376 OA90 9090 9090 52E8 ; Offset 0F0
MIX1	B800 008E C026 803E 3C03 7775 095F 5E59 ; Offset 02E
MIX1-2	B800 008E C0BE 7103 268B 3E84 0083 C70A ; Offset 02A
New Zealand (1)	0400 B801 020E 07BB 0002 B901 0033 D29C ; Offset 043
New Zealand (2)	0400 B801 020E 07BB 0002 33C9 8BD1 419C ; Offset 041
Palette	EB2B 905A 45CD 602E C606 2506 0190 2E80 ; Offset ?, 1538 bytes
Pentagon	8CC8 8ED0 BC00 F08E D8FB BD44 7C81 7606 ; Offset 037
South African 1	1E8B ECC7 4610 0001 E800 0058 2DD7 00B1 ; Offset 158
South African 2	1E8B ECC7 4610 0001 E800 0058 2D63 00B1 ; Offset 158
Spanish	E829 068E E005 B419 CD21 8884 E300 E8CE ; Offset ?
Surviv 1.01	0E1F B42A CD21 81F9 C407 721B 81FA 0104 ; Offset 304, 897 bytes
Surviv 2.01	81F9 C407 7228 81FA 0104 7222 3C03 751E ; Offset 05E, 1488 bytes
Surviv 3.00	03F7 2E8B 8D15 00CD 218C C805 1000 8ED0 ; Offset 0B0, 1813 COM, 1808 EXE
Syslock	8AE1 8AC1 3306 1400 3104 4646 E2F2 5E59 ; Offset 0, 3551 bytes
Traceback	B419 CD21 89B4 5101 8184 5101 8408 8C8C ; Offset 104, 3066 bytes
Typo	5351 521E 0656 0E1F E800 005E 83EE 24FF ; Offset 01D, 867 bytes
Vienna (1)	8BF2 83C6 0A90 BF00 01B9 ; Offset 005, 648 bytes
Vienna (2)	FC8B F281 C60A 00BF 0001 B903 00F3 A48B ; Offset 004, 648 bytes
Vienna (3)	FC89 D683 C60A 90BF 0001 B903 00F3 A489 ; Offset 004
Virus-90	558B 2E01 0181 C503 0133 C033 BBB9 0900 ; Offset 01E
Yale	BB40 008E DBA1 1300 F7E3 2DE0 078E C00E ; Offset 009
Yankee	E800 005B 81EB D407 2EC6 875C 00FF FC2E ; Offset 0
Zero Bug	81C9 1F00 CD21 B43E CD21 5A1F 59B4 43B0 ; Offset 100

Reported only

4K	E808 OBE8 D00A E89A OAE8 F60A E8B4 0A53 ; Offset 239
Agiplan	E9CC 0390 9090 9090 9C50 31C0 2E38 26DA ; Offset 0 (?)
Amstrad	C706 0E01 0000 2E8C 0610 012E FF2E 0E01 ; Offset 114
December 24th	C606 7E03 FEB4 5290 CD21 2E8C 0645 0326 ; Offset 044
Ghostballs	AE75 EDE2 FA5E 0789 BC16 008B FE81 C71F ; Offset 051
MachoSoft	5051 56BE 5900 B926 0890 DIE9 8AE1 8AC1 ; Offset ?
Oropax	06B8 E033 CD21 3CFF 7423 8CCE 8EC6 8B36 ; Offset ?
Perfume	FCBF 0000 F3A4 81EC 0004 06BF BA00 57CB ; Offset 0AA
Swap	31C0 CD13 B802 02B9 0627 BA00 01BB 0020 ; Offset ?
Sylvia	CD21 EBFE C3A1 7002 A378 0233 COA3 9E02 ; Offset 229

AIDS patterns - not a virus

REM$.EXE	4D5A OC01 1E01 0515 6005 0D03 FFFF 3D21 ; Offset 0
AIDS.EXE	4D5A 1200 5201 411B E006 780C FFFF 992F ; Offset 0

VIRUS DISSECTION

Fridrik Skulason

Disk Killer

The ‘Disk Killer’ virus, also known as ‘Ogre’ virus, was first reported in June 1989 in the United States but has recently appeared in other countries including the UK. It is a boot sector virus which infects hard-disks as well as diskettes. An infected computer will function normally until the virus ‘triggers’, which happens if the computer is left running for at least 48 hours*. The virus will then encrypt everything found on the bootable partition of the hard disk, making the disk non-bootable in the process. The first reports of this virus were rather inaccurate, saying that the virus was "virtually undetectable by present means", but in fact it is no harder to detect and eradicate this virus than other similar viruses such as ‘Brain’, ‘Italian’ and ‘New Zealand’.

Operation

Like any other boot sector virus, Disk Killer can only infect a computer if it is booted from an infected diskette. The first part of the virus is stored on the boot sector. This has two main functions. One is to make room for the virus in memory, by decrementing the amount of available memory by 8K. This will make a 640K computer to appear to contain only 632K. The second function of this part is to load and execute the main body of the virus, which is stored elsewhere on the diskette.

It will then hook into two interrupts, INT 8H (timer tick) and INT 13H (disk I/O). The original INT 8H and INT 13H vectors are moved to INT 81H and INT 83H respectively.

Finally, the virus will load and execute the original boot sector.

Every time the INT 8H routine is called (which normally occurs 18.2 times per second) the virus will increment a 24-bit counter by one. This counter is used by the INT 13H routine to determine if the destructive part of the virus should be triggered.

The INT 13H routine will only activate in the case of a disk read command (AH = 2). In other cases it will simply issue an INT 83H command, which transfers control to the original INT 13 routine.

When a disk read command is given, the virus will first check if he counter described above has reached 300000 (hex). If so, the destructive part of the virus will be activated. Starting from zero, the counter will reach 300000 (hex) 48 hours after the computer is turned on. However, the counter is not only incremented on a timer tick. If the virus has already infected the boot sector of a hard disk and the computer is then booted from a diskette, the value of the counter will be added to the value stored on the hard disk.

Disk Killer will then check if an attempt is being made to read the boot sector of a diskette. If so, it will check if the diskette is already infected. If not, it the virus will try to infect it. This means that most diskettes inserted in an infected machine will become infected.

If the computer is booted from an infected diskette and an attempt is later made to read from a hard disk, the virus will try to infect the hard disk.

The virus uses a two byte signature to mark the boot sectors it infects. If it finds the values CB 3C in locations 3E and 3F on the boot sector, it is assumed to be already infected. This makes it possible to ‘inoculate’ against the virus. Be careful not to inoculate bootable diskettes in this way, however.

On a floppy, the virus will hide by using a method similar o that used by the Italian virus. It searches for an unused 6K block to hide in and mark the corresponding entries as ‘bad’ in both copies of the FAT.

Hard disks are infected in a different way. Instead of hiding in sectors marked as ‘’bad’, the virus will search for free space in the ‘hidden’ sectors between the partition boot record and the first sector of the first partition.

**Editor’s note: Network file servers are normally left powered on permanently and are thus particularly vulnerable to this virus.*

The Virus will encrypt everything found on the bootable partition of the hard disk....

The Attack

When the destructive part of the virus is activated it will display the following messages on the screen:

```
Disk Killer – Version 1.00 by COMPUTER OGRE
04/01/1989
Warning!!
Don’t turn off the power or remove the
diskette while Disk Killer is Processing
```

The word ‘PROCESSING’ (in capitals) also appears in the centre of the screen.

The virus then scrambles all the data on the bootable partition of the hard disk. It will read one track at a time into memory, encrypt it, using a simple XOR algorithm and then write it back. The first track subjected to this attack is track 0, which contains the partition boot record. Encrypting it will make the disk non-bootable and reading from it may cause a ‘general failure’ error message.

In the case of a floppy-only machine, the virus will just scramble the diskette in drive A:.

When the disk scrambling process is finished, the virus will display the message:

```
Now you can turn off the power.
I wish you luck !
```

The computer will then start executing an infinite loop, forcing the user to switch the machine off.

Should the virus activate on your computer, you have two choices. One is to turn the computer off as soon as possible in the hope that the encrypted part if the disk can be restored. This should be possible, unless the virus has started scrambling the second copy of the FAT. The other is to allow the process to continue and then seek a restoration program. Remember, the current version of the virus does not destroy the information stored on the disk, it only encrypts it.

Detection

Since the virus produces no obvious effects until it triggers, it is harder to detect than the Italian or Brain viruses. The best way to ascertain infection is to search for the signature the virus uses to mark infected boot sectors (see above) or to search the boot sector for the hexadecimal string

```
2EA1 1304 2D08 002E A313 04B1 06D3 E08E
```

starting at location 195 (C3 hex).

Disinfection

Removing the virus from an infected diskette or hard disk is a reasonably straightforward process. The only thing that needs to be done is to locate the original boot sector and write it back in its original position. This can be done by scanning through all clusters marked as bad, searching for a readable boot sector. In the case of an infected hard disk, the original boot sector should be located somewhere in the first sectors of the disk.

Final Note

Disk Killer is the first boot sector virus which does not assume a sector size of 512 bytes. This means that the number of sectors the virus occupies in not always the same, since sometimes a sector size of 1024 bytes is used.

Our acknowledgements to Fridrik Skulason for supplying descriptions and hexadecimal patterns of a large number of recent IBM PC viruses (see page 11).

We should also like to thank Yisrael Radai for his amendments to the *Virus Bulletin Table of Known IBM PC Viruses* which will appear next month.

VIRUS REPORT

David Ferbrache

WDEF - The Hidden Virus

On December 3rd 1989 a new Macintosh virus was reported. The WDEF virus was detected by one of its known symptoms - the crashing of 2Ci systems when modifying an active diskette window on the desktop.

The virus spreads by infecting the desktop file which is maintained in the disk root directory. This file normally contains information on the placement of folder/ application icons (in the form of resources). In the case of a WDEF infection this file will also contain a single "WDEF 0" resource.

The WDEF (window definition resource) contains executable code to create or modify a window on the Mac. Such code can carry out a variety of functions on invokation (such as replicating in a viral manner).

Symptoms of the virus include:

· Crashes of 2Ci and portables

· Crashes under multifinder when saving application files to disk

· Erroneous display of outline front styles

· Performance problems on AppleShare servers

· Presence of a WDEF resource in the desktop

Two versions have been detected: WDEF A (detected in France, Belgium and throughout the USA) and WDEF B (a version which beeps on infection of the desktop - detected only in Ohio, USA).

The virus is widespread which indicates that it had been spreading un-noticed for a prolonged period.

Desktop Resources

In finder the desk top resource file is opened subsequent to the system resource file, and thus due to the first-in-last-out search order will replace resources appearing in the system file. Thus the standard "WDEF 0" resource is replaced by the viral desktop "WDEF 0".

The desktop of the disk can be rebuilt (removing the WDEF virus if the active system is uninfected) by holding down the COMMAND and OPTION keys as the system is rebooted or when a new disk is inserted.

Evasion from Detection

Existing detection utilities (both scan applications and protection INITs) will not detect the virus.

The virus contains code to bypass monitoring of resource manager calls by protection INITs. This is achieved by resetting the resource manager trap vectors to point to the ROM routines during the execution of the viral code, and then to restore previous trap vectors.

For this purpose the virus carries hard coded copies of the pointer to *AddResource, ChangedResource, UpdateResFile, WriteResource* and *PBWrite* for a variety of ROM types. These pointers are used (by *SetTrapAddress*) to reset the vector to ROM.

Removal

Rebuilding the desktop will remove the virus from an infected disk. A preferable approach is to obtain:

· Disinfectant 1.5 (released 14 December 1989)

· Gatekeeper Aid 1.0 (a small INIT designed to supplement the Gatekeeper protection INIT, can be used with the SAM intercept)

Specific removal utilities such as Eradicator are also available in the public domain.

The addition of a line

```
Creator - ERIK & Resource WDEF & Any
```

to search string for Virus Detective 3.0 is recommended.

In Summary

· WDEF spreads by infecting the invisible desktop file on each disk

· The virus is spread by transfer of disks between machines not by individual application or data files

· Disks with no visible files (which are formatted) can be infected

This virus demonstrates some radical departures from previous Macintosh viruses and a more thorough analysis will appear in a later edition of Virus Bulletin (Ed.).

WORM PROGRAMS

David Ferbrache

Wide and Local Area Network Worms Part 1

Unix Worms - Potential and Fact

This article is the first of a series describing the operation (both of actual and potential) Wide and Local Area Network worms under various operating systems and network protocols. The first of these is the Unix operating system using uucp, UK Joint Academic NETwork (JANET) and Internet protocols.

A working knowledge of networking under System V or Berkeley Software Distribution (BSD) 4.2/3 is assumed.

Introduction

Worms replicate without a host program by copying their code from one host system to another on a network. To do this the worm must breach system security on the target host, or utilise trusted paths which exist between the two hosts.

Four principal methods exist:

1. Circumventing access controls by search over password or access code domains

2. Circumventing trusted host controls by impersonation

3. Utilising bugs in utilities comprising the applications layer of the network interface

4. Interception and replay of messages including passwords and access codes

A combination of method (1) and (3) was utilised by the Internet worm in addition to its use of the BSD rlogin trusted host system

Applications Layer Interface

The programs comprising the application layer interface (layer 7 in the OSI communications model) to the Unix operating system depend on the underlying protocol stack, for the three protocols discussed they are:

INTERNET PROTOCOLS

ftp - File transfer protocol.
tftp - Trivial file transfer protocol.
smtp - Simple mail transfer protocol.
nntp - Network news transfer protocol.
telnet - Remote terminal protocol.
rsh, rlogin, rcp, rexec - BSD 4.2/3 trusted host "r" protocols for command execution, login, file transfer and remote procedure call.
A number of minor protocols managed by the Internet daemon program (inetd), including fingerd.

JOINT ACADEMIC NETWORK

Protocols:

Blue book - File transfer protocol
Grey book - Electronic mail protocol
Red book - Job transfer and manipulation protocol
Green book - x29 terminal access protocol

UUCP protocols:

uucp - Unix-Unix file transfer protocol
uux - Unix remote command execution protocol

Proprietary:

rcp - Sun remote procedure call protocol
nfs - Sun network filing system protocol

Each program comprising the applications layer interface must be secure in order to prevent violation of host security. Regrettably, this is not the case at the moment. A number of known bugs are described (in brief) later in this paper.

Password Search

The *telnet*, *rlogin* and *Green book* protocols allow access over the network to the host by presenting an interface similar to a direct terminal login. A user code and password will be prompted for before allowing login to a standard or restricted command shell.

When shell access is obtained, by use of a suitable user code and password, the remote end of the worm can copy itself using a variety of encoding techniques for tranfer of binary information (eg *uuencode*) to a file on the remote system. The originating system will then send a command sequence to run (where necessary compiling and linking) the transferred worm program.

Unix passwords are stored in encrypted form in a publicly readable file (*/etc/passwd*). Encryption is by a modified Data Encryption Standard (DES) algorithm, applied 25 times. A standard constant is encrypted by a 48 bit key derived from the user password. The algorithm is perturbed by a pseudorandom seed based on the system clock at time of encryption. This perturbation is designed to frustrate hardware search using DES chips sets.

On a typical system it is possible to break a password by exhaustive search within 1 minute. This is achieved by encrypting all words in the Unix on line directory and comparing the encrypted version with that in the password file.

Precautions such as preventing users from selecting passwords appearing in the standard dictionary, which are pronounceable (formed from standard trigrams), related to the gcos (name information) field in the password file or the user's account name. The use of shadow password files (storing passwords in secure form) and password ageing (expiring passwords after a fixed time), used in BSD 4.3 and System V respectively, also reduce the likelihood of a password being cracked.

It is worth noting that the encryption code in the Internet worm was well written and considerably faster than the default encryption routine supplied with Unix.

Trusted Hosts

The Berkley software distribution (BSD 4.2/3) includes four *"r"* protocols. These protocols operate by allowing login from trusted hosts without additional authentication. As such while convenient they represent a significant relaxation of access control.

The trusted hosts are specified in two groups of files:

system wide lists of hosts (*/etc/hosts.equiv*)
user specific lists of hosts (*.rhosts* in each home directory)

Unfortunately due to the ability of each user to specify additional hosts which he/she regards as trusted, there are often complex trusted paths via numerous hosts and user ids which can result in full (*root*) privileges.

The trusted host system was utilised by the Internet worm to obtain remote shells on peer systems.

A serious flaw also exists in the *rsh/rlogin* service daemon (a program which services requests from users and runs in the background) on certain Unix systems. This bug is due to failure to check that the correspondent in any communication is a *rsh* or *rlogin* command (specifically that the originating port id is not privileged, ie < 1024). Thus it is possible for a connection to be made by a worm program which appears to originate from a *rsh/rlogin* running under a trusted usercode (*root, bin, daemon*), thus allowing worm code to be transported and run at a permission which will allow full system privileges to be gained.

Normally *bin* or *daemon* user ids will yield root access with minimal effort - either by replacement of a setuid utility in a *bin* writeable directory, or by manipulation of the "*crontab*" or "*at*" delayed command execution spool areas.

Know Bugs in the Network Interface

Bugs are known to exist in the following utilities (all of which run with root permission). It is impossible without testing to identify which vendor patches have been issued (vendors are often very slow to identify and correct security problems).

tftp	Failure to test for access permission on intermediate directories in destination file paths, coupled with the writeable */etc/utmp* error.
smtp	Debugging mode enabled; bug in code to inhibit posting by root to file paths rather than recipient user.
fingerd	*ugh*. Buffer bound check omitted. Plan file permission check omitted.
rh/rlogin	Trusted path originating port id check omitted.
uucp	Shell meta character screening defective allowing command substitution on the remote system.
rexec	Authentication flawed.
walld	Defect terminal path checking coupled with theutmp bug
ftp	

Furthermore, bugs are known to exist in commands which execute with root permission (set uid) which when utilised will allow anyone to gain full access, including

login	-P option authentication defective
chfn	gcos field size check defective
passwd	Resource limit error return handled incorrectly
at	Spool area and file permission incorrect
restore	Erroneously distributed setuid
getconf	Permission checking on sub program execution flawed
mem/kmem	Device permissions often incorrect

The fingerd bug utilised by the Internet worm is now well known, and will serve to give an indication of the subtlety of bugs in the interface.

A common problem in Unix is that of the input string over-running the target buffer. The original gets command does not check for array bounds when the string length is excessive, this causes the string to overwrite data following the array or worse (if the array is on the runtime stack), corrupting or modifying the stack return frame.

The fingerd did not use the bound checking fgets library routine, and thus could be forced by careful choice of input request to overwrite the stack return so that the main command would exit by calling the command shell.

This command shell remained attached to the input and output of the fingerd command, which is attached to the distant system which invoked the finger service. Thus a shell with permission equal to that at which fingerd was running has been invoked (normally root).

A Typical Worm Attack

A typical attack sequence would be:

a. Probe remote system using smtp, fingerd, rsh and uucp bugs to obtain root (case 1,2), user (3) or uucp (4) permissions

b. In cases 3 and 4 exploit one of the known bugs to upgrade permission to root (login, chfn, passwd or at bugs)

c. Copy the worm code to the remote system

d. If source code then compile and link for execution

e. Run the worm on the remote system

Camouflage and Concealment

The remote worm would normally exploit a range of concealment techniques including:

1. Forking at regular intervals to change process id and minimise accumulated system time.

2. Change argv[0] command name shown by ps to an innocent name (eg sh, ps, ls etc)

3. Trap all signals (especially SIGQUIT which causes a core dump)

4. Minimise processor load by careful use of waits

5. Target execution to periods of high load, inactivity or overnight

6. Encrypt all recognisable strings in the executable file

The Internet worm applied methods 1, 2 and 6; but failed to minimise processor activity thus forcing the load on infected systems beyond 20. This caused immediate overload and a system crash. While a denial of service attach was caused, this was apparently not the intention of the author.

Interception and Replay

A final security loophole which can be exploited is the ability to intercept broadcast and Internet point-point links. The former by direct monitoring software (such as the SunOS Network Interface Tap [NIT], the latter by the use of ICMP (Internet Control Message Protocol) to cause routing table updates, resulting in the forwarding or rerouting of packets to the compromised host.

Passwords transiting network links in clear (ie supplied as part of a telnet/x29 login or as arguments to su/passwd during a rlogin session) can easily be intercepted if any host on the network is compromised by a worm.

Finally, communication packets which have been intercepted can often be re-injected to forge an id at a latter time. This is trivial in the case of User Datagram Protocol (UDP) based services but complex in the case of class 4 transport protocols such as Transmission Control Protocol (TCP).

Propagation of Information and Remote Control

A worm could be designed to compromise both access codes (by transmitting clear or encrypted passwords) or information (using *root* access, or by *fingerd* or *smtp* bugs by user access) to a remote host. Traffic padding could be added to conceal the data flow, as could reduction and randomisation of data transfer rates.

Finally, in the case of a high security system the end destination could be concealed by covert channel techniques (eg by varying the routing of packets by Address Resolution Protocol (ARP) table update it is possible to communicate additional bits of information which can be detected by a host monitoring packet flow through the comprimised host).

To consider the reverse problem, it is possible for a centralised authority to reconfigure an active network worm by sending control packets. Such techniques could allow on-the-fly incorporation of new attack techniques and known system ids and passwords. This would also allow concentration of effort by network node.

Distributed worms would also be expected to exchange information to avoid duplication of effort and to perform global network load balancing (ironically the reason for the original experiments with network worms).

The Internet worm incorporated a mechanism to send a single byte back to an Arpanet host presumably to monitor spread. The worm also included simple methods to detect multiple instances of a worm on a single host and to arbitrate termination.

Conclusion

The potential threat from worm programs is significant. I feel that it is only a matter of time before a recurrence of the Internet worm incident occurs. In all likelihood the next worm will utilise improved concealment techniques and will not exhibit the obvious overloading effects (which enabled rapid detection and elimination) that the previous worm did.

OPINION

Phil Crewe

Should We Trust Public Domain Anti-Virus Software?

Before answering that question, I think it's sensible to treat public domain software and anti-virus software as separate issues. Each deserves consideration on its own, and an answer to the question should not even be contemplated until all the relevant facts (and fantasies) are in their true place. Please note that in all reference to public domain software, I also include shareware. I take this stance for the very simple reason that with reference to anti-virus packages, most users will use these terms interchangeably.

Are there any intrinsic differences between commercial and public domain software?

Naturally there are some differences between these two approaches to software, even excepting the purchasing differences! This is because generally the author of a shareware package has written the software in response to a particular demand, and because it concerns a topic of interest to him. The other important point is that the author is usually a one-man band. This does lead to a certain 'type' of software being released into the public domain or shareware market place. In general commercial software will do more that a public domain package. This is true for anti-virus software. I must admit, however, that this is less true in other areas of shareware software; one example is telecommunications, where shareware packages are usually better than their commercial counterparts. If we specially consider anti-virus software here, then one would typically expect to get two or three public domain packages for a complete anti-virus toolkit (for example a scan program, a disinfector, and a memory resident 'watcher'), where the commercial package would probably comprise all three items in one kit. It may well be true that these are separate programs, but they should all work together and be supported from a single source. This is another major difference between the types. With a commercial package you can (generally) rely on software support via telephone with a single company who are either the developers or the distributors of the product. You can expect them to be capable of answering your questions, and would rightly complain if they could not. With a public domain package, however, the author may be reachable, but probably only by electronic mail to a remote Bulletin Board, and therefore response times are usually longer. Also one has to consider that a significant proportion of these authors have other jobs, and therefore development time is somewhat more restricted.

You should expect good documentation with the commercial application. I say this with tongue-in-cheek as this is probably the exception rather than the rule! Commercial documentation should (in theory) be written and presented more professionally than that provided with public domain software, although I have to say that most public domain releases come nowadays with excellent documentation. The final difference which should be mentioned is that (notwithstanding the time availability point I made earlier) the update response of good public domain software is generally better than the commercial products. This is because a reputable author will release updates quite frequently, and these are circulated throughout the Bulletin Board community with great speed, hence users tend to get updates easily and quickly. After relating all of these differences, I should also say that they are in many ways similar. Both are trying to do exactly the same job, even if they do it in different ways. The difference is almost *moral* - many software authors feel that they wish to produce software for the 'public good' rather than profit. It should be noted, though, that some types of product lend themselves more easily to public domain releases, and therefore good public domain software tends to be in certain well-defined area. Luckily anti-virus toolkits are one such area.

Is there any difference between anti-virus software and software of any other type?

The answer here must be a resounding **no!** All software should be dealt with in exactly the same way, be it spreadsheet, DTP package, anti-virus, or, for that matter, a game. All software should be obtained from a reputable source. This could be the local specialist centre, or a good Bulletin Board. Providing the source is reputable, that is the first, and indeed I would suggest the main, criterion. Once software has been obtained, it should be checked before putting it into full circulation or use. This should point out any clashes with other operating software, and anything in the running of the software to be wary of. Remember one thing - there are many so-called Trojans in circulation, and a plethora of badly written software which can cause untold problems on a particular hardware platform, or when running with other software. One man's invaluable utility is another man's weekend spent rebuilding a disk! It must be said that this is more true of public domain than commercial software. After all, in the commercial sector bad publicity means lower sales, so at least the basics of the software are tested. The one thing that most users do not do is find out what the package is meant to do before running it. This is especially true of public domain software, where the name of the program might look interesting, so it is run to find out what it does. The simple task of reading the documentation would prevent many disasters. Also, if you can see the software running somewhere else, do take a look before installing it on your machine. You can also use this contact for help and advice (both before and during use). The final thing that I will say about dealing with new software is really something that you should do whether or not your software changes at all. **BACKUP** regularly. At least if you have good up-to-date backups then any new piece of software which causes problems will not cause irreplaceable loss.

Should we be generally wary of public domain software?

This really depends on the software and the source of supply. I am always wary of public domain software if (a) *I do not know either the software purports to do*, and (b) *I do not know where it came from*. All new public domain anti-virus packages (or any other public domain packages for that matter) gets treated with a vast amount of caution when it first arrives, but no more so than new commercial software. If the software is from a reputable source, and if it arrives with decent documentation, contacts for getting additional information or help, and with all the files present which are supposed to be present (and with the correct modification dates of that information is given), then generally I take it to the next stage of testing. If it does not comply with these points, then I would be wary and I would take steps to obtain more information before running it. My situation is somewhat different from a general user, though. I tend to get all sorts of software sent my way and, running a Bulletin Board, I see a lot of public domain and shareware programmes. When asked for an opinion on good anti-viral software, I always indicate both commercial and shareware products which will do the required job which the circumstances dictate. This is because I feel that both types of packages do the same job, and do it well.

So, should we trust public domain anti-virus software?

The only answer I can give to that is "*as much as we trust any other anti-virus package*". All new anti-virus products have (by their very nature) to be treated with some respect. Some would day that the best way of circulating a virus (especially one with a long lead time before the 'punchline') is in an anti-virus package. However, this is also true for any other software! I have seen my fair share of viruses in commercial (shrink wrapped) packages, and there is no real reason to suppose that more viruses exist in public domain or shareware software than in commercial software. It remains, however, easier to circulate a virus in the public domain, which is why it is essential to obtain software from reputable sources. It doesn't matter at the end of the day whether that is a reputable distribution or dealer, or a reputable Bulletin Board System. I must also say that (in my experience) some of the best programmes are public domain or shareware. One example is *Disinfectan*t for the Macintosh, which is probably all-round as effective as any other product on the market, and the update speed is excellent. A new version is usually released within days of detection and isolation of a new virus. It definitely has the lead on commercial software in this regard, as the lead times here are more like two to three months.

One last thought. Since (hopefully) obtaining anti-virus software is a precaution rather than a necessity, a user may never know exactly how effective the measures he is taking are. It is therefore extremely important to rely on reports and evaluations of product from people who have tested it against a wide range of know threats. I trust public domain and shareware utilities, but I temper this trust with testing and recommendations. I would not use public domain software blindly, and if it was totally unknown I would find out as much about it as possible before running it. If all else fails, read the documentation and leave a message for the author requesting the functionality and latest version numbers of the programme. If this is not forthcoming, or if there is no way to contact anyone else about it, it is immediately despatched to the big bit-bucket in the sky, with a staple through the disk. There are also one or two bits of commercial software I would like to do this to, possibly for different reasons…

NEWS

A Computer Virus Bibliography, believed to be the first, has been compiled by Jack Bologna, Associate Professor of Management at Siena Heights College in Michigan, USA. It contains listing of books, journals, research papers, newspaper and magazine articles concerning computer viruses since early 1988. It is available free of charge by writing the Professor Jack Bologna, Siena Heights College, 1247 E. Siena Heights Drive, Adrian, Michigan 49221, USA.

The Computer & Security **Computer Virus Handbook** edited by Dr. Harold Joseph Highland is now available. The 375 page volume contains a wealth of information about computer viruses affecting IBM PCs, with details of defensive measures and evaluations of some twenty anti-virus products. The book costs £85 an is available from Elsevier Advanced Technology, Mayfield House, 256 Banbury Road, Oxford OX2 7DH, England, UK. Tel 0865 512242. *Virus Bulletin* will publish a review in February.

A two day **seminar** entitled **Computer Viruses** has been organised by SAL (State of the Art Ltd) and will take place in four European cities (Frankfurt, Milan, Paris, London) in March. The presenter is Dr. Douglas Tygar. Assistant Professor of Computer Science, Carnegie Mellon University and member of the US. Defense Advanced Research Project Agency (DARPA) and computer Emergency Response Team (CERT), the seminar aims to address the vulnerabilities of a variety of operating systems and to analyse methods to counter virus and worm attacks. Details from SAL CV Seminar, Victoria House, Suite M9, Southampton Row, London WC1B 4EF. Tel 01 404 3341.
Fax 01 405 6203.

VIRUS BULLETIN

Subscription price for 1 year (12 issues) including delivery:

US$ for USA (first class airmail) $350, Rest of the World (first class airmail) £195

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, Haddenham, Aylesbury, HP17 8JD, England
Tel (0844) 290396, International Tel (+44) 844 290396
Fax (0844) 291409, International Fax (+44) 844 291409

US subscriptions only:

June Jordan, Virus Bulletin, PO Box 875, 454 Main Street, Ridgefield, CT 06877
Tel 203 431 8720, Fax 203 431 8165