

COMPUTER VIRUSES

Note The lecture on computer viruses (Gresham College, London, 6pm 21 February 2002) will include discussion of real viruses, details of how they work, and discussion of how they may be tackled. These notes provide some background, history, and further information.



Computers can do anything: from running spreadsheets, word processors, power stations to music synthesisers and missile control systems. And because computers can do anything, they can in particular run viruses and any other nasty software. Of course viruses aren't the only way computer systems can go wrong: you can have your computer stolen; you might simply have a disc crash or some other hardware fault. None of these problems are predictable.

However, viruses are unique in their abilities. They can stop many computers at once: this would be *much* more serious for a small company than 'normal' faults that affect only one PC at a time. Thus, viruses rank with hazards like power cuts and fire in their effect and speed of action. Worse than fire, you may find that you cannot take your work elsewhere, for if you did you might simply take the virus infection with you and bring more systems down too! Secondly, viruses can spread disinformation and bring disrepute to individuals or organisations: viruses may send malicious email apparently on behalf of the person whose computer has been infected.

Normal information management precautions, such as taking backups, can be of considerable help in avoiding and recovering after virus attacks. (But you may also have backed up your viruses.)

Definition A *computer virus* is a piece of program code that attaches copies of itself to other programs, incorporating itself into them so that the modified programs, while still possibly performing their intended function, surreptitiously do other things. Programs so corrupted seek others to which to attach the virus, and so the "infection" spreads. Successful viruses lie low until they have thoroughly infiltrated the system, and only reveal their presence when they cause damage. The effect of a virus is rarely traceable back to its originator, so viruses make attractive weapons for vandals.

Computer viruses generally work by altering files that contain otherwise harmless programs. This is *infection*. When an infected program is invoked, it seeks other programs stored in files to which it has write permission, and infects them by modifying the files to include a copy of itself (*replication*) and inserting an instruction to branch to that code at the old program's starting point. Then the virus starts up the original program so that the user is unaware of its intervention. (The exact details vary from virus to virus.)

A virus can spread when information is shared: on a multi-user system with shared disk facilities, or on a PC where users download programs from bulletin boards, or if they share floppy disks or other exchangeable media. Email is a common vector, especially when powerful email programs run arbitrary programs 'to be helpful' — but which turn out to be able to run viruses.

If person A executes one of person B's programs (or programs embedded in email, *etc*) that is infected, A's programs and files risk becoming infected, since programs that A invokes normally have permission to alter A's own files. Even when they never execute one another's programs, infection can spread from A to B through an intermediary. In practice, it is hard to guard against infection in an environment that encourages program (data or email...) sharing. Even in an isolated personal computer environment, viruses easily spread from one floppy disk to another once having infiltrated the system.

Many applications interpret their data in some way as programs. Spreadsheets, email programs, and word processors typically have "macro programs," such as Visual Basic, to provide extra functionality. Viruses can therefore infect even the data which is handled by such applications. Viruses can be spread, then, by users sharing a word processor or spreadsheet file: it is not necessary to share a 'program' as such.

Other Malicious Programs The term "virus" is a popular catch-all for other kinds of malicious software. A *logic bomb* or *time bomb* is a destructive program activated by a certain combination of circumstances, or on a certain date. A *Trojan horse* is any bug inserted into a computer program

that takes advantage of the trusted status of its host by surreptitiously performing unintended functions. A *worm* is a distributed program that invades computers on a network. It consists of several processes or “segments” that keep in touch through the network; when one is lost (e.g., by a server being rebooted), the others conspire to replace it on another server — they search for an ‘idle’ server, load it with copies of themselves. Like viruses, worms spread by replication; unlike them, they may run as independent processes rather than as part of a host program.

It is unproductive to be pedantic over the precise terms; the main distinction is that viruses replicate themselves, and Trojan horses explicitly deceive users. Of course, viruses may include Trojan components, and often do.

To escape detection, viruses normally reside in compiled programs rather than as readable source code and thus do not survive recompilation or re-installation of a back-up. Viruses in scripting languages (e.g., Visual Basic) use encryption for the same reason: to try to conceal their purpose from casual readers, and indeed from virus detection programs. Just as worms are destroyed by simultaneously rebooting all affected machines, viruses are eradicated by simultaneously recompiling all affected programs. However, under special circumstances a bug can survive recompilation: in a compiler that is written in the language it compiles (a common bootstrapping practice), it is possible to implant a bug that re-inserts itself into the binary code whenever the compiler is recompiled. Such a bug need never be visible in source form.

History and Examples In 1962 V. A. Vyssotsky, a researcher at Bell Telephone Laboratories (New Jersey), devised a game called Darwin, played on an IBM 7090 computer. The object of the game was survival, for “species” of programs to fight it out in an arena set up in the 7090’s memory. Successful programs would be able to make more copies of themselves, and perhaps go on to take over the entire arena. The idea was taken up ten years later (Aleph Null, 1972), and converted to run on other computers. Within the somewhat arbitrary rules of the game, Doug McIlroy invented what he called a virus: an unkillable organism that could still fight.

The idea, though, of a *maliciously* self-propagating computer program originated in Gerrold’s 1972 novel *When Harlie Was One*, in which a program called telephone numbers at random until it found other computers into which it could spread. Worms were also presaged in science fiction, by Brunner’s 1975 novel *The Shockwave Rider*. The first published report of a worm program, done in 1982 by Shoch and Hupp as an experiment in distributed computing, included a quotation from this book.

The idea of self-replication and viral *ideas* was explored by Douglas Hofstadter (1983), partly taking up the ideas of Richard Dawkin’s *memes*: ideas that persist (like genes) because they replicate (Dawkins, 1976). Hofstadter claimed the ideas of self-reproducing ideas could be traced back to 1952. The *Scientific American* published Hofstadter’s articles, but was probably first to popularise viruses in the computer sense with Dewdney’s series of articles on a game called corewars (over the period 1984–87). Corewars is a game where players wrote attacking programs, not unlike today’s viruses in theory, but the programs all ran within corewars rather than running wild on a PC unrestricted.

The first actual virus program seems to have been created in 1983, as the result of a discussion in a computer security seminar, and described at the AFIPS Computer Security Conference the following year.

It is interesting that when Roberto Cerruti and Marco Morocutti read Dewdney’s corewars articles they thought out how to write a virus for the Apple II computer, and considered installing a disc at the biggest computer shop in Bresica (Italy) to cause an epidemic in their home city. They thought that a ‘real’ epidemic had to be malignant, and that after 16 reproductive cycles they would have the virus reinitialise (erase) discs. Having realised how devious their idea was they decided not to do it, but instead told the world via Dewdney’s column in the *Scientific American*.

Dewdney went on to consider spreading advertising copy by virus: the infected computer could display irritating messages in commercial breaks! “It’s time you got DISC DOCTOR, available at a store near you.” The idea of using virus-like mechanisms for information dissemination was later pursued by Thimbleby and Witten (1990) in a system called Liveware.

In 1984 Ken Thompson, in his Turing Award Lecture, showed how a self-replicating bug can infest a compiler or other language processor and become completely invisible — an idea mentioned above.

Real virus attacks were not reported until a few years thereafter, and so far have been more in the nature of electronic vandalism than serious subversion. One of the first occurred in late 1987

when, over a two-month period, a virus quietly insinuated itself into IBM-PC programs at a Jerusalem university. It was noticed because it caused programs to grow longer (due to a bug, it repeatedly reinfected files). Once discovered, it was analysed and an antidote devised. It was designed to slow processors down on certain Fridays, and to erase all files on Friday, 13 May.

At about the same time, another PC virus invaded Lehigh University, and a much-publicised “chain letter” Christmas message spread itself by self-replication, clogging the Bitnet network. The latter was eradicated only by a massive network shutdown. Early 1988 saw a relatively harmless Macintosh virus designed to distribute a “message of peace,” and a number of other viruses appeared for this and other PCs. By that time talk about viruses had invaded the news media.

At 9pm on 2 November 1988, a worm program was inserted into the Internet computer network by Cornell graduate student Robert Morris, Jr. It exploited several security flaws in systems running Unix to spread itself from system to system. Although discovered within hours, it required a huge effort (estimated at 5,000 hours and \$200,000) by programmers at affected sites to counteract and eliminate it over a period of weeks. It was unmasked by a bug: under some circumstances it replicated itself so fast that it seriously slowed down the infected host. On 21 January 1990, Morris was convicted and in May 1990 he was sentenced to three years probation and fined \$10,000. In addition, he was ordered to perform 400 hours of community service.

On 6 March 1992, the Michelangelo virus (so named because 6 March is Michelangelo’s birthday), although widely heralded as a worldwide threat to computer systems, actually did little damage.

Viruses now have no distinct identity, but (typically) “mutate” each time they copy themselves to other files. This, combined with various cryptographic techniques, makes modern viruses difficult to detect. Some viruses have “stealth” code, and behave differently when a user attempts to detect them. False alarms have become an increasing problem, particularly with users sending “chain email” warning about supposed virus problems; ironically, the panic may cause more problems than the viruses it warns about!

Email has become a popular way to distribute viruses, because powerful commercial email packages (e.g., Microsoft Outlook) are themselves programmable — and users often configure email systems to ‘helpfully’ run programs automatically! If it doesn’t happen automatically, the virus mail typically tricks the user into running its program, which then reads the user’s email address list and mails out further copies of itself to the user’s contacts. Once installed, the *payload* part of the virus may do nasty or innocuous things, depending on the whim of the virus writer, such as deleting files on a specific date.

Viruses are not difficult to develop, and many ‘virus construction kits’ are readily available on the internet. Many viruses are simple variants of others, indicating that virus writers are often unimaginative — but no less dangerous for that.

Computer viruses and real viruses Despite their similarities, computer viruses do not behave like biological viruses. For example, Koch’s Postulates (which apply to biological agents; cf. Thimbleby, Anderson & Cairns, 1999) are irrelevant. Computer viruses are purely conceptual, and indeed the equivalent of computer viruses for the human mind — as opposed to digital computers — are called *memes*. Memes are potentially either good or bad, but for any supposedly ‘good’ computer virus there is always a far better way of achieving the same ends without using a virus (and without the risk of giving a vandal a virus model to copy and subvert).

Computer viruses have stimulated the research field of “artificial life,” the study of what non-carbon based life might be like, and of course mostly life as simulated on computer. In contrast to viruses, which are destructive, artificial life has a positive outlook! See Thimbleby, Witten and Pullinger (1995) for a review of the differences, which are beyond the scope of these notes.

Defences The obvious, but generally impractical, defence against viruses is never to use anyone else’s software and never to connect with anyone else’s computer — assuming you could guarantee that the computer never had an infection to start with. One can achieve this using cryptographic techniques. Another defence is to implement a check in the operating system that queries users whenever a program they have invoked attempts to write to disk. In practice, however, this imposes an intolerable burden because users do not generally know what files their software does legitimately. Given a particular virus, one can write an *antibody program* that spreads itself in the same way, removing the original virus from infected programs, and ultimately removing itself too. However, this approach cannot protect against viruses in general, since it is not possible to tell whether a particular piece of code is a virus or not — worse, some

such antibodies have been 'hijacked' by virus writers and been turned to be destructive. There is at least one example of a virus writer (belatedly) releasing anti viral software to detect his own virus: Vaxene was an anonymous program to detect Scores infections (which it did not do very well) but it claims to have been written by the Scores author.

Digital signatures can help prevent the corruption of files. Each file as it is written is sealed by attaching an encrypted checksum. Also, before it is used, the checksum is decrypted and checked against the file's actual checksum. Such a scheme may engender unacceptable overhead, however, in the logistics of handling encryption keys. Moreover, it also assumes that the file is not infected *before* it is signed, and that the checking system itself has not been compromised (e.g., by a Trojan horse).

A practical approach is to regularly (or continuously) run programs that recognise (enough) viruses, and which try to eliminate virus infections before they do too much damage. Because new viruses are being devised every day, it is important to keep detection programs up to date (e.g., by regular subscription from a reputable company), and to minimise risky procedures (e.g., sharing information as infrequently as possible).

All approaches are trade-offs. The only real hope is eternal vigilance on the part of users, and, above all, education of users to the possible consequences of their actions.

Theoretical problems The basic problem of detecting a virus (or Trojan) is that there are an infinite number of ways of getting a computer to do anything. Consider printing the number 5. You could ask the computer to print 5, or $4+1$, or $3+2$, or $100-95$, or $10\sin(\pi/6)$. Only the first few of these alternatives is readily recognisable as printing 5; and, as the last example shows, you can make recognition that the program prints 5 as difficult as you wish.

So, *even supposing* that there was exactly one sort of virus, there would still be an infinite number of ways of doing the same nasty work — and most of those ways would be unrecognisable. In fact, there are an infinite number of ways in which viruses might work, and each of these ways can itself be expressed in an infinite number of ways. Actual viruses don't really take advantage of the infinite possibilities, and in fact it would not be possible to do so!

Advice Be very careful opening unsolicited email with attachments, especially if you are using Microsoft Outlook or similar programmable products. Never download software from web sites unless you have antivirus software to check it. **Your organisation should have policies**, including the following points:

Have 'fire drills'

If your organisational rules permit it, run 'fire drills' where you simulate the impact of a virus (or other computer problems).

Awareness

Just as you wouldn't share food (goodness knows what diseases you might catch!) with someone you don't know very well, don't share programs (files, emails...). Just as you wouldn't eat food you found lying on the ground, don't run any old program that comes your way. Disable all automatic programs, such as automatically running email attachments. And so on! Remember: computer viruses work very much like real viruses.

Use more than one hardware platform

If you have the resources, one of the best protections against total disaster is to use several different sorts of computer (for instance, PCs and Macintoshes). In many cases you will be able to run compatible software on a range of equipment (if your software is so good then it should be available on many machines). Very few viruses can migrate from one machine type to another, though some viruses embedded in (for example) spreadsheet files may be able to if you are running the same spreadsheet package on different systems.

Further reading

- 2001 Thimbleby, H., Anderson, S. O. & Cairns, P. "Reply to 'Comment on "A Framework for Modelling Trojans and Computer Virus Infection"' by E. Mäkinen," *Computer Journal*, **44**(4):324–325.
- 1999 Thimbleby, H., Anderson, S. O. & Cairns, P., "A Framework for Modelling Trojans and Computer Virus Infection," *Computer Journal*, **41**(7):444–458.
- 1995 Thimbleby, H., Pullinger, D. J. & Witten, I. H., "Concepts of Cooperation in Artificial Life," *IEEE Transactions on Systems, Man & Cybernetics*, **25**(7):1166–1171.

- 1994 Thimbleby, H., "An Organisational Solution to Piracy and Viruses," *Journal of Systems and Software*, **25**(2):207–215.
- 1994 Cohen, F. B., *A Short Course on Computer Viruses*, 2nd. ed., New York: John Wiley.
- 1991 Hafner, K. & Markoff, J., *Cyberpunk: Outlaws and Hackers on the Computer Frontier*, New York: Simon and Schuster.
- 1991 Thimbleby, H., "Can Viruses Ever Be Useful? *Computers and Security*, **10**(2):111–114.
- 1990 Witten, I. & Thimbleby, H., "The Worm That Turned", *Personal Computer World*, 202–206, July.
- 1989 Spafford, E. H., "The Internet Worm: Crisis and Aftermath," *American Scientist* (June).
- 1989 Stoll, C. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, New York: Doubleday.
- 1984 Dewdney, A. K., "Computer Recreations," *Scientific American*, May 1984, 15–19; March 1985; September 1985.
- 1983 Hofstadter, D. R., "On Viral Sentences and Self-Replicating Sentences", *Scientific American*, January, 14–19. (Reprinted in *Metamagical Themas*, Basic Books, 1985).
- 1982 Shoch, J. F. & Hupp, J. A., "The 'Worm' Programs—Early Experience with a Distributed Computation," *Communications of the ACM*, **25**:172–180.
- 1976 Dawkins, R., *The Selfish Gene*, Oxford University Press (2nd. edition, 1989).
- 1972 Aleph Null (⌘), "Computer Recreations," *Software—Practice and Experience*, **2**:93–96.

(These notes are much revised from **Virus, Computer** by H. Thimbleby & I. H. Witten, in *The Encyclopedia of Computer Science*, edited by A. Ralston, E. D. Reilly & D. Hemmendinger, editors, 4th. ed., pp1839–1841, Nature Publishing Group, 2000.)