The Internet Worm

- Compromising the availability and reliability of systems through security failure

What happened

- In November 1988, a program was deliberately released that spread itself throughout Digital VAX and Sun workstations across the Internet. It exploited security vulnerabilities in Unix systems.
- In itself, the program did no damage but it's replication and threat of damage caused extensive loss of system service and reduced system responsiveness in thousands of host computers.
- This program has become known as the Internet Worm.
- This was the first widely distributed Internet security threat.

Terminology

- A worm
  - This is a program that can autonomously spread itself across a network of computers.
- A virus
  - This is a program that can spread itself across a network of computers by attaching itself to some other program or document.
- A trapdoor
  - This is a vulnerability in a program that allows normal security controls to be bypassed.

Consequences of the worm

- Strange files appeared in systems that were infected.
- Strange log messages appeared in certain programs.
- Each infection caused a number of processes to be generated. As systems were constantly re-infected, the number of processes grew and systems became overloaded.
- Some systems (1000s) were shut down because of the problems and because of the unknown threat of damage.

Worm description

- Program was made up of two parts
  - A main program that looked for other machines that might be infected and that tried to find ways of getting into these machines;
  - A vector program (99 lines of C) that was compiled and run on the infected machine and which then transferred the main program to continue the process of infection.
- Security vulnerabilities
  - fingerd - an identity program in Unix that runs in the background;
  - sendmail - the principal mail distribution program;
  - Password cracking;
  - Trusted logins.

fingerd

- Written in C and runs continuously.
- C does NOT have bound checking on arrays. fingerd expects an input string but the writer of the worm noticed that if a longer string than was allowed for was presented, this overwrote part of memory.
- By designing a string that included machine instructions and that overwrote a return address, the worm could invoke a remote shell (a Unix facility) that allowed priviledged commands to be executed.

sendmail

- sendmail routes mail and the worm took advantage of a debug facility that was often left on and which allowed a set of commands to be issued to the sendmail program.
- This allowed the worm to specify that information should be transferred to new hosts through the mail system without having to process normal mail messages.

Password cracking

- Unix passwords are encrypted and, in the encrypted form, are publicly available in /etc/passwd.
- The worm encrypted lists of possible passwords and compared them with the password file to discover user passwords.
- It used a list of about 400 common words that were known to be used as passwords.
- It exploited fast versions of the encryption algorithm that were not envisaged when the Unix scheme was devised.

Trusted logins

- On Unix, tasks can be executed on remote machines.
- To support this, there is the notion of a trusted login so that if a login command is issued to machine Z from user Y in machine X then Z assumes that X has carried out the authentication and that Y is trusted; no password is required.
- The worm exploited this by looking for machines that might be trusted. It did this by examining files that listed machines trusted by the current machine and then assumed reciprocal trust.

Killing the worm

- The main effects of the worm were in the US and system managers worked for several days to devise ways of stopping the worm.
- These involved devising modifications to the programs affected so that the worm could not propagate itself, distributing these changes, installing them then rebooting infected machines to remove worm processes.
- The process took several days before the net was cleared of infection.

What we learned

- Security vulnerabilities result from flaws and these will always be with us. Problems can be fixed but new problems can arise with new versions of software.
- Diversity is good - we need a heterogeneous not a homogeneous network.
- Isolationism is not the answer - those sites that disconnected from the network were amongst the last to resume service.

The perpetrator

- The perpetrator was a student at Cornell University.
- He was discovered fairly quickly and charged.
- His sentence was for a period of community service and a $10, 000 fine
    - This was relatively light as the major thrust of his defence was that the program explicitly did not cause deliberate damage and, in fact, he had tried (but failed) to ensure that too many processes would not be generated on host machines.

Warning

- Students before and since this infection have been curious about security and have written experimental programs. Few of these students are wicked and many of them are very competent programmers.
- However, the consequences of experiments that go wrong are now so great that network authorities do not distinguish between stupidity and malice. There are severe penalties for any experiments that compromise system security.

Finding out more

- Communications of the ACM, 32 (6), June 1989 has a number of articles on the Internet worm.
- Computer-related Risks. P. G. Neumann, Addison Wesley 1995. A compendium of information about system failures that have compromised safety, security and reliability.
- See Intranet web pages for links.