IN-61-CR

280667

P-16

The Internet Worm

Peter J. Denning

7 Feb 89

RIACS Technical Report TR-89.3

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

The Internet Worm

Peter J. Denning

7 Feb 89

RIACS Technical Report TR-89.3

NASA Cooperative Agreement Number NCC 2-387

The Internet Worm

Peter J. Denning

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report TR-89.3
7 Feb 89

In November 1988 a worm program invaded several thousand UNIX-operated Sun workstations and VAX computers attached to the Research Internet, seriously disrupting service for several days but damaging no files. An analysis of the worm's decompiled code revealed a battery of attacks by a knowledgeable insider, and demonstrated a number of security weaknesses. The attack occurred in an open network, and little can be inferred about the vulnerabilities of closed networks used for critical operations. The attack showed that password protection procedures need review and strengthening. It showed that sets of mutually trusting computers need to be carefully controlled. Sharp public reaction crystalized into a demand for user awareness and accountability in a networked world.

This is a preprint of the column *The Science of Computing* for
American Scientist 77, No. 2 (March-April 1989).

Work reported herein was supported in part by Cooperative Agreement NCC 2-387
between the National Aeronautics and Space Administration (NASA)
and the Universities Space Research Association (USRA).

The Internet Worm

Peter J. Denning

Research Institute for Advanced Computer Science

7 Feb 89

Late in the evening of 2 November 1988 someone released a "worm" program into the ARPAnet. The program expropriated the resources of each invaded computer and generated replicas of itself on other computers, but did no apparent damage. Within hours, it had spread to several thousand computers attached to the worldwide Research Internet.

Computers infested with the worm were soon laboring under a huge load of programs that looked like innocuous "shell" programs (command interpreters). Attempts to kill these programs were ineffective: new copies would appear from Internet connections as fast as old copies were deleted. Many systems had to be shut down and the security loopholes closed before they could be restarted on the network without reinfestation.

Fortuitously, the annual meeting of UNIX experts opened at Berkeley on the morning of November 3. They quickly went to work to capture and dissect the worm. By that evening, they had distributed system fixes to close all the security loopholes used by the worm to infest new systems. By the morning of November 4, teams at MIT, Berkeley,

and other institutions had decompiled the worm code and examined the worm's structure in the programming language C. They were able to confirm that the worm did not delete or modify files already in a computer. It did not install Trojan horses, exploit superuser privileges, or transmit passwords it had deciphered. It propagated only by the network protocols TCP/IP, and it infested only computers running Berkeley UNIX but not AT&T System V UNIX. As the community of users breathed a collective sigh of relief, system administrators installed the fixes, purged all copies of the worm, and restarted the downed systems. Most hosts were reconnected to the Internet by November 6, but the worm's effect lingered: a few hosts were will disconnected as late as November 10, and mail backlogs did not clear until November 12.

The worm's fast and massive infestation was so portentous that the *New York Times* ran updates on page one for a week. The *Wall Street Journal* and *USA Today* gave it front-page coverage. It was the subject of two articles in *Science* magazine (*1,2*). It was covered by the wire services, the news shows, and the talk shows. These accounts said that over 6,000 computers were infested, but later estimates put the actual number between 3,000 and 4,000, about 5% of those attached to the Internet.

On November 5 the *New York Times* broke the story that the alleged culprit was Robert T. Morris, a Cornell graduate student and son of a well-known computer security expert who is the chief scientist at the National Computer Security Center. A friend reportedly said that Morris intended no disruption; the worm was supposed to propagate slowly but a design error made it unexpectedly prolific. When he realized what was happening, Morris has a friend post on an electronic bulletin board instructions telling how to disable the worm -- but no one could access them because all affected computers were

down. As of February 1989, no indictments had been filed against Morris as authorities pondered legal questions. Morris himself was silent throughout.

The worm's author went to great lengths to confound its discovery and analysis, a delaying tactic that permitted the massive infestation. By early December 1988, Eugene Spafford of Purdue (*3*), Donn Seeley of Utah (*4*), and Mark Eichin and Jon Rochlis of MIT (*5*) had published technical reports about the decompiled worm that described the modes of infestation and the methods of camouflage. (See Box 1.) They were impressed with the worm's battery of attacks, saying that, despite errors in the source program, the code was competently done. The National Computer Security Center requested them and others not to publish the decompiled code, fearing that troublemakers might reuse the code and modify it for destructive acts. Seeley replied that the question is moot because the worm published itself in thousands of computers.

The reactions of the computer science community have been passionate. Some editorial writers report that Morris has become a folk hero among students and programmers, who believe that the community ought to be grateful that he showed us weaknesses in our computer networks in time to correct them before someone launches a malicious attack. The great majority of opinion, however, seems to go the other way. Various organizations have issued position statements decrying the incident and calling for action to prevent its recurrence. No other recent break-in has provoked similar outcries.

The organization Computer Professionals for Social Responsibility issued a statement calling the release of the worm an irresponsible act and declaring that no programmer can guarantee that a self-replicating program will have no unwanted consequences. The statement said that experiments to demonstrate network vulnerabilities should be

done under controlled conditions with prior permission, and it called for codes of ethics that recognize the shared needs of network users. Finally, the statement criticized the National Computer Security Center's attempts to block publication of the decompiled worm code as short-sighted because an effective way to correct widespread security flaws is to publish descriptions of those flaws widely.

The boards of directors of the CSNET and BITNET networks issued a joint statement deploring the irresponsibility of the worm's author and the disruption in the research community caused by the incident. Their statement called for a committee that would issue a code of network ethics and propose enforcement procedures. It also called for more attention to ethics in university curricula. (At Stanford, Helen Nissenbaum and Terry Winograd have already initiated a seminar that will examine just such questions.)

The advisory panel for the division of networking and research infrastructure at NSF endorsed the CSNET/BITNET statement, citing as unethical any disruption of the intended use of networks, wasting of resources through disruption, destruction of computer-based information, compromising of privacy, or actions that make necessary an unplanned consumption of resources for control and eradication. The Internet Activities Board has drafted a similar statement. The president of the Association for Computing Machinery called on the computer science community to make network hygiene a standard practice (*6*). A congressional bill introduced July 1988 by Wally Herger (R-Calif.) and Robert Carr (D-Mich.), called the Computer Virus Eradication Act, will doubtless reappear in the 101st Congress.

Obviously, all this interest is provoked by the massive scale of the worm's infestation and the queasy feeling that follows a close call. It also provides an opportunity to

review key areas of special concern in networking. In what follows, I will comment on vulnerabilities of open and closed networks, password protection, and responsible behavior of network users.

The rich imagery of worms and viruses does not promote cool assessments of what actually happened and of what the future might hold. It is interesting that as recently as 1982 worm programs were envisaged as helpful entities that located and used idle workstations for productive purposes (*7*); most people no longer make this benign interpretation. Some of the media reports have mistakenly called the invading program a virus rather than a worm. A virus is a code segment that embeds itself inside a legitimate program and is activated when that program is; it then embeds another copy of itself in another legitimate but uninfected program, and it usually inflicts damage (*8*). Because the virus is a more insidious attack, the mistaken use of terminology exaggerated the seriousness of what happened. Given that the security weaknesses in the Internet service programs have been repaired, it is unlikely that an attack against these specific weaknesses could be launched again.

While it is important not to overestimate the seriousness of the attack, it is equally important not to underestimate it. After all, the worm caused a massive disruption of service.

It is important to aknowledge a widespread concern that grew out of this attack: Are networks on which commerce, transportation, utilities, national defense, space flight, and other critical activities depend also vulnerable? This concern arises from an awareness of the extent to which the well-being of our society depends on the continued proper functioning of vast networks that may be fragile. When considering this question, it is

important to bear in mind that the Internet is an open network and the others are closed.

What is the risk to an open network? Because the Internet is open by design, its computers also contain extensive backup systems. Thus, in the worst case, if the worm had destroyed all the files in all the computers it invaded, most users would have experienced the loss of only a day's work. (This contrasts starkly to the threat facing most PC users, who because of the lack of effective backup mechanisms stand to lose years of work to a virus attack.) In addition, users would certainly lose access to their systems for a day or more as the operations staff restored information from backups.

What are the implications for other networks? Computers containing proprietary information or supporting critical operations are not generally connected to the Internet; the few exceptions are guarded by gateways that enforce strict access controls. For example, the Defense Department's command and control network and NASA's space shuttle network are designed for security and safety; it is virtually impossible for a virus or worm to enter from the outside, and internal mechanisms would limit damage from a virus or worm implanted from the inside. Given that the Internet is designed for openness, it is impossible to draw conclusions about closed networks from this incident.

Calls to restrict access to the Internet are ill-advised. The openness of the Internet is closely aligned with a deeply held value of the scientific community, the free exchange of research findings. The great majority of scientists are willing to accept the risk that their computers might be temporarily disabled by an attack, especially if a backup system limits losses to a day's work.

The next area that calls for special concern is password security. Although trapdoors and other weaknesses in Internet protocols have been closed, password protection

is a serious weakness that remains. (See Box 2.) The risk is compounded by "mutually trusting hosts," a design in which a group of workstations is declared as a single system: access to one constitutes access to all.

Many PC systems store passwords as unenciphered cleartext, or they do not use passwords at all. When these systems become part of a set of trusting hosts, they are an obvious security weakness. Fortunately, most systems do not store passwords as cleartext. In UNIX, for example, the login procedure takes the user's password, enciphers it, and compares the result with the user's enciphered entry in the password file. But one can discover passwords from a limited set of candidates by enciphering each one and comparing it with the password file until a match is found. One study of password files revealed that anywhere from 8% to 30% of the passwords were the literal account name or some simple variation; for example, an account named "abc" is likely to have the password "abc", "bca", or "abcabc" (*9*). The worm program used a new version of the password encryption algorithm that was nine times faster than the regular version in UNIX; this allowed it to try many more passwords in a given time and increased its chances of breaking into at least one account on a system. Having broken into an account, the worm gained easy access to that computer's trusted neighbors.

The final area of special concern is the responsibilities of people who participate in a large networked community. Although some observers say that the worm was benign, most say that the disruption of service and preemption of so many man-hours to analyze the worm was a major national expense. Some observers have said that the worm was an innocent experiment gone haywire, but the experts who analyzed the code dispute this, saying that the many attack modes, the immortality of some worms, and the elaborate

camouflage all indicate that the author intended the worm to propagate widely before it was disabled. Most members of the computer science community agree that users must accept responsibility for the possible wide-ranging effects of their actions and that users do not have license to access idle computers without permission. They also believe that the professional societies should take the lead in public education about the need for responsible use of critical data now stored extensively in computers. Similarly, system administrators have responsibilities to take steps that will minimize the risk of disruption: they should not tolerate trapdoors, which permit access without authentication; they should strengthen password authentication procedures to block guessed-password attacks; they should isolate their backup systems from any Internet connection; and they should limit participation in mutually trusting groups.

Certainly the vivid imagery of worms and viruses has enabled many outsiders to appreciate the subtlety and danger of attacks on computers attached to open networks. It has increased public appreciation of the dependence of important segments of the economy, aerospace systems, and defense networks on computers and telecommunications. Networks of computers have joined other critical networks that underpin our society -- water, gas, electricity, telephone, air traffic control, banking, to name a few. Just as we have worked out ways to protect and ensure general respect for these other critical systems, we must work out ways to promote secure functioning networks of computers. We cannot separate technology from responsible use.

References

1. E. Marshall. 1988. "Worm invades computer networks." *Science* **242** (11 Nov 1988). 855-856.

2. E. Marshall. 1988. "The worm's aftermath." *Science* **242** (25 Nov 1988). 1121-1122.

3. E. Spafford. 1988. "The Internet worm program: an analysis." Technical Report No. CSD-TR-823, available from Computer Sciences Department, Purdue University, W. Lafayette, IN 47907. Published in the ACM *Computer Communication Review*, January 1989, available from ACM, Inc., 11 W. 42 St., New York, NY 10036.

4. D. Seeley. 1988. "A tour of the worm." Technical Report available from the Computer Science Department, University of Utah, Salt Lake City, UT 84112. In *Proc. Winter Usenix Conf.*, February 1989, Usenix Association.

5. M. Eichin and J. Rochlis. 1988. "With microscope and tweezers: an analysis of the Internet Virus of November 1988." Technical Report, MIT Project Athena, Cambridge, MA 02139.

6. B. Kocher. 1989. "A hygiene lesson." *Communications of ACM* **32**, 3 & 6.

7. J. F. Shoch and J. A. Hupp. 1982. "The worm programs -- early experience with a distributed computation." *Communications of ACM*, **25**, 3. 172-180.

8. P. J. Denning. 1988. "Computer viruses." *American Scientist* **76**, 3 (May-June). 236-238.

9. F. T. Grammp and R. H. Morris. 1984. "UNIX operating system security." *AT&T Bell Labs Technical Journal* **63**, 8. October. 1649-1672.

How the worm worked

The Internet worm of November 1988 was a program that invaded Sun 3 and VAX computers running versions of the Berkeley 4.3 UNIX operating system containing the TCP/IP Internet protocols. Its sole purpose was to enter new machines by bypassing authentication procedures and to propagate new copies of itself. It was prolific, generating on the order of hundreds of thousands of copies among several thousand machines nationwide. It did not destroy information, give away passwords, or implant Trojan horses for later damage.

A new worm began life by building a list of remote machines to attack. It made its selections from the tables declaring which other machines are trusted by its current host, from users' mail-forwarding files, from tables by which users give themselves permission for access to remote accounts, and from a program that reports the status of network connections. For each of these potential new hosts, it attempted entry by a variety of means: masquerading as a user by logging into an account after cracking its password; exploiting a bug in the finger protocol, which reports the whereabouts of a remote user; and exploiting a trapdoor in the debug option of the remote process that receives and sends mail. In parallel with attacks on new hosts, the worm undertook to guess the passwords of user accounts on its current host. It first tried the account name and simple permutations of it, then a list of 432 built-in passwords, and finally all the words from the local dictionary. An undetected worm could have spent many days at these password-cracking attempts.

If any of its attacks on new hosts worked, the worm would find itself in communication with a "shell" program -- a command interpreter -- on the remote machine. It fed that shell a 99 line bootstrap program, together with commands to compile and execute it, then broke the connection. If that bootstrap program started successfully, it would call back the parent worm within 120 seconds. The parent worm copied over enciphered files containing the full worm code, which was compiled from a C program containing about 3,000 lines. The parent worm then issued commands to construct a new worm from the enciphered pieces and start it.

The worm also made attempts at population control, looking for other worms in the same host and negotiating with them which would terminate. However, a worm that agreed to terminate would first attack

many hosts before completing its part of the bargain -- leaving the overall birthrate higher than the deathrate. Moreover, one in seven worms declared itself immortal and entirely bypassed any participation in population control.

The worm's author went to considerable pains to camouflage it. The main worm code was enciphered and sent to the remote host only when the bootstrap was known to be operating there as an accomplice. The new worm left no traces in the file system: it copied all its files into memory and deleted them from a system's directories. The worm disabled the system function that produces "memory dumps" in case of error, and it kept all character strings enciphered so that, in case a memory dump were obtained anyway, it would be meaningless. The worm program gave itself a name that made it appear as an innocuous shell to the program that lists processes in the system, and it frequently changed its process identifier.

Protecting Passwords

The worm's dramatic demonstration of the weakness of most password systems should prompt a thorough examination in the context of networks of computers. The following are basic desiderata:

1. Every account should be protected by a password.
2. Passwords should be stored in an enciphered form, and the file containing the enciphered passwords should not be publicly accessible (it is in UNIX).
3. Passwords should be deliberately chosen so that simple attacks cannot work -- for example, they could include a punctuation mark and a numeral.
4. New passwords should be checked for security -- many systems have (friendly!) password checkers that attempt to decipher passwords by systematic guessing, sending warning messages to users if they are successful.
5. To make extensive guessing expensive, the running time of the password encryption algorithm should be made high, on the order of one second. This can be achieved by repeatedly enciphering the password with a fast algorithm.
6. New cost-effective forms of user authentication should be employed, including devices to sense personal characteristics such as fingerprints, retinal patterns, or dynamic signatures, as well as magnetic access cards.
7. Sets of computers that are mutually trusting in the sense that login to one constitutes login to all need to be carefully controlled. No computer outside the declared set should have unauthenticated access, and no computer inside should grant access to an outside computer.