NISTIR 4545

Computer Security: Selected Articles

Marianne Swanson
NIST Editor

Elizabeth B. Lennon
NIST Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 4545

Computer Security: Selected Articles

Marianne Swanson
NIST Editor

Elizabeth B. Lennon
NIST Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

April 1991

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Table of Contents

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY INTERNAL REPORT

COMPUTER SECURITY: SELECTED ARTICLES

INTRODUCTION

This National Institute of Standards and Technology Internal Report (NISTIR) presents nine articles which represent a wide spectrum of computer security information. The articles were selected by the staff of the Computer Security Division, Computer Systems Laboratory, at the National Institute of Standards and Technology. The information provided is by no means comprehensive; rather the articles offer a quick reference or an introduction to a specific security technology. The article "Is your System Safe?" is a high-level overview of the Internet worm and addresses ways to correct various system vulnerabilities. The article "Computer Viruses and Personal Computers" provides guidance on preventing and handling computer viruses. The remaining articles discuss software copying, local-area-network security, computer ethics, data security responsibilities, risk analysis, and encryption. This publication will benefit computer security managers as well as managers and users of information technology.

The second part of this document contains a reading list prepared by the Information Exchange Working Group of the Computer and Telecommunications Security Council, a government/industry technical group that was sponsored by NIST from 1987 to 1990. The list provides titles, sources, and abstracts of important computer security publications that were relevant to the council's interests.

The National Institute of Standards and Technology (NIST) makes no claim or endorsement of the information provided. However, as this material may be of use to other organizations, NIST is reprinting the articles with permission as part of a continuing effort to assist federal agencies in accordance with NIST's mandate under the Computer Security Act of 1987.

Questions regarding this publication should be addressed to National Institute of Standards and Technology, Computer Security Resource Center, A-216 Technology, Gaithersburg, MD 20899. Additional copies of this publication may be purchased through the National Technical Information Service, Springfield, VA 22161, telephone: (703) 487-4650.

IS YOUR SYSTEM SAFE?

A year after a worm brought the Internet to its knees, the danger to UNIX networks still exists

By Frank Hayes

Early this year, Cornell University graduate student Robert T. Morris was convicted of creating a rogue computer program—a "worm"—and releasing it into the Internet. When this issue of UNIXWORLD went to press, Morris was still awaiting sentencing. But the evidence that convicted Morris brought home the stunning impact of the Internet's collapse.

In all, the worm temporarily disabled as many as 3000 machines on the network, which links UNIX-based computers at universities, businesses, and military research facilities.

IS YOUR SYSTEM SAFE?

It took days of intensive work (8000 personnel hours, by one estimate) to isolate the worm and clear it from the Internet. No data was destroyed, but realistic estimates of time lost on the affected machines range as high as $10 million.

And, astonishingly, more than a year later some of the bugs that Morris' worm program exploited still threaten UNIX users.

Could It Happen Again?

The Internet worm used three major means of attack. Two involved flaws in the mail system of Berkeley extensions to UNIX: a problem in the `sendmail` program, and another in `fingerd`. The third was a password-guessing routine that gave the worm

It took days of intensive work (8000 personnel hours, by one estimate) to isolate the worm and clear it from the Internet.

access to other Berkeley system utilities it used to propagate itself. Because it depended on these flaws, the worm only succeeded in infecting computers running Berkeley versions of UNIX and its derivatives. It was also limited to certain kinds of hardware—DEC VAXes running BSD UNIX, and some Sun workstations. (See Editor-at-Large Rik Farrow's article describing how the worm did its work.)

Quick fixes for the problems with `sendmail` and `fingerd` became available almost immediately after the worm incident from the teams of programmers at Berkeley and MIT who disassembled the worm. And new versions of the software have since been released by the vendors, plugging the security holes.

But according to UNIX security experts we talked to, some systems on the Internet still haven't upgraded their software—and far more non-Internet systems still have the old versions of these programs.

That's a serious problem, according to Beverly Ulbrich, Sun Microsystems' product manager for SunOS security. Forcing users to upgrade their software is impossible. "Obviously, the most we can do is make people aware of the problem," Ulbrich says. The `sendmail` problem is a case in point. Sun distributed a fixed version a year ago, but Ulbrich says bug reports continue to arrive at Sun—complaining about security problems with the old version. "It's one of the biggest frustrations we've got," she says.

Hoping to change the situation, Ulbrich says, Sun is currently developing a program to streamline the process of converting bug reports into corrected software that's actually on users' machines. But it's not as simple as sending mailgrams to every Sun owner. "When we hear about a problem, we can't just immediately publicize it. We have to make sure we have a workaround or new binaries," says Ulbrich.

Then there's the problem of distributing the fixes once they're available. Ulbrich says Sun is considering dial-up machines, as well as backup telephone and fax systems that could be used when a rampaging worm has forced system administrators to disconnect their computers from the outside world. Sun hopes to have a pilot system in place this summer, she adds.

Mail Call

Even when users know new versions are available, they're sometimes reluctant to install them—especially if the users have changed the program at the source-code level to add new features. Ulbrich isn't convinced that's a real reason for not upgrading: "It's a real

According to UNIX security experts, some systems on the Internet still haven't upgraded their software.

excuse that I've heard," she says. But at least some users disagree.

Peter G. Neumann, a computer scientist at SRI International who also moderates an on-line network conference on computer risks to society, says he isn't happy with the version of `sendmail` he's got—but he doesn't

---

PC Viruses, UNIX Worms

A worm is a standalone computer program that reproduces itself. That's in contrast to a virus, which is a program section that can copy itself onto other programs. Viruses are a widespread problem on desktop microcomputers, including IBM PC-compatibles and Apple Macintoshes. Virus programs on these PCs are notorious for corrupting data, erasing files, and reformatting hard disks. By contrast, the Internet worm—the most devastating worm program to hit the UNIX community—merely wasted thousands of hours of computer time.

Why do viruses strike PCs, while worms hit UNIX networks? PC viruses are typically spread on floppy disks; when an infected program on the disk is run, the virus copies itself onto programs on other floppy or hard disks. Viruses are common on PCs because of the standardized hardware (which allows virus code to run without recompilation), and because virtually all PCs use floppy disks as their primary way of getting software into the computer.

UNIX machines as a group share neither of these risks—but UNIX networks allow different avenues of attack. For example, the highly developed mail system that links many UNIX systems allows unscreened messages to enter your system—and, in some cases, unscreened program code. The Internet worm exploited a common bug in the `sendmail` utility that allowed a program to be sent as a message, then compiled and executed on the system that received it.

In addition, most PCs can only run one program at a time—so to be executed, the virus code must link itself to an application someone is likely to use. But multitasking UNIX systems allow worms to make copies of themselves that run independently—and without any legitimate user's knowledge.

As a result, while PCs are at risk from relatively passive viruses that are carried from one computer to another on floppy disks, UNIX machines are more likely to be attacked by worms that actively mail themselves to other computers, attempt to guess passwords, and reproduce themselves as widely as possible.

—F.H.

IS YOUR SYSTEM SAFE?

want to upgrade to a new "safe" version because it lacks some of the functionality he needs.

Nor does Neumann believe a new version of the program will really be secure from attack. "You have to remember, the problems in `sendmail` have been in there forever," says Neumann. The Internet worm depended on `sendmail` having been compiled with the debugging option included, but, says Neumann, "It's not just the debug option. It used to be that anybody could connect to it and gain superuser status."

Part of the problem, Neumann explains, is that UNIX was designed not only with little concern for security, but originally, with little concern for mail, either. "In 1965 there was a solution to the `sendmail` problem in Multics," he says. In Multics, the mail system could only append mail messages to the mailbox—it couldn't read files or have any other access to the system. But when Ken Thompson created UNIX as a scaled-down version of Multics, he didn't include the Multics mail system because he didn't need mail—UNIX was to be a single-user system. When mail was later added in the Berkeley extensions to UNIX, the implementers used superuser—the source of most of the `sendmail` security problems.

Reminding users about upgraded vendor software can be handled with e-mail bulletins.

"Think about what you're trying to do," says Neumann. Getting what he describes as "good solutions with a little bit of robustness to them" requires intelligent management and cooperation among all the users of the many UNIX mail networks—and it will still only work within reasonable requirements. "If you want a wide open exchange between everyone in the world, it's a problem. It's a very difficult problem," says Neumann.

The Password Problem

But at least as much of a problem as buggy utilities is poor passwords. The Internet worm included a password-guessing routine that allowed it to worm its way onto at least some machines. And in general, passwords are the system of choice for humans trying to break into computers, UNIX-based or otherwise.

The Computer Emergency Response Team (CERT) was set up shortly after the Internet worm incident to monitor problems that could threaten the network. The CERT Coordination Center, at Carnegie-Mellon University's Software Engineering Institute in Pittsburgh, collects reports of computerized break-ins and other security problems from users, and regularly issues advisories to system administrators.

Are things more secure since the worm attack? "They're better, yes, in the sense of knowing what kinds of activities are going on," says Terry McGillen, a spokesman for CERT. But though CERT keeps better tabs on security problems today (it runs a 24-hour hotline at 412-268-7090), the actual number of break-ins seems to be rising.

According to McGillen, there are two major sources of problems: break-ins that guess passwords and those that exploit holes in vendors' software. Reminding users about upgraded vendor software can be handled with e-mail bulletins; improving passwords is the tougher problem to deal with. "People know they shouldn't do this, but they've got too many things to remember," says McGillen, "so they use their zodiac sign, or their birthdate, or their social security number as a password." Worse still, users will sometimes put that same easy-to-remember password on *all* their computer accounts.

Just Your Ordinary Joe

Russell Brand, a former government computer security expert, believes password security is so lax on most systems that even moderate work to tighten things up will discourage most hackers. In a primer he's developing on dealing with computer security problems, Brand points out that the most common case of a poor password is what he calls the "Joe," an account where the password is the same as the user name.

Making the user name the password makes the password easy to remember—and exceptionally easy to guess. Brand says he has never found a system that didn't have at least one Joe. Last summer, he says, "While I was testing a series of sensitive systems where hundreds of thousands of dollars were spent to remove security

UNIX Self-Defense

Can your system be infiltrated? You can cut down the risks dramatically by tightening your policies in just two areas: passwords and software updates.

Easy-to-guess passwords are the biggest problems on most systems. Watch out for:

- A password that matches the account name.
- A password of "password," "passwd," or no password at all.
- The same password used on many different machines.
- Easily guessed passwords such as common names.

To cut password risks, you should:

- Use automated checks for bad passwords.
- Keep password files from being readable by users.
- Regularly purge dormant accounts, which frequently have simple passwords.
- Make sure your users know the importance of choosing a good password and keeping it secret—and how to change the password.
- Discourage users from leaving for the day without logging off.

Software updates will continue to be a problem, although more standardized versions, such as System V Release 4, may help. In the meantime:

- Check regularly with software suppliers for updates and bug fixes.
- If your staff has modified utilities, document the modifications as completely as possible.
- Make bug fixes—especially for security bugs—a priority.

Most of all, beware of the common attitude that security precautions on UNIX systems are a waste of time. Open systems don't *have* to be open to attack.

—*F.H.*

IS YOUR SYSTEM SAFE?

holes, including rewriting a fair fraction of the operating system, there were Joes."

There are simple ways to check for Joes and other bad passwords, and to encourage users to pick (and protect) their passwords more carefully. Unfortunately, says Brand, many system administrators simply believe that system security is a waste of time.

Sun's Beverly Ulbrich sees the same attitude. Some companies, she says, "don't care at all. But it gets important as soon as people get hit with a problem. Somebody guesses a password or finds a guest account, and the system gets broken into...and then they panic."

There are simple ways to check for bad passwords, and to encourage users to pick (and protect) their passwords more carefully.

Some system administrators do seem to take a casual attitude toward security. Scott Todd, system administrator at Cadre Technologies in Beaverton, Ore., has never had a break-in. He admits that "I probably run the least secure network of any of the system administrators I know," and says of security procedures, "they waste time and they waste other people's time." But Todd carefully screens new software that's introduced into his network of 40 Sun workstations, and because his network is only a mail site, he knows he's protected from attacks such as the Internet worm. And though he takes no special precautions for password security, he says, "We're a small site; as we get bigger I may have to pay more attention to that."

Another system administrator, who asked not to be named, is currently building a much larger UNIX system at a major financial services company, and he's substantially more concerned with security. Like Todd, this administrator says: "Internet is not an issue with us. This is going to be a commercial network, so ours is going to be pretty tightly wrapped."

A bigger concern is modems that will connect the six to eight file servers and hundreds of PC workstations across the country: "The modem is our first line of defense. If it's off, it's off," he says. And password security will be backed up by physical security for the computer sites, which will be

How the Worm Turned

On Wednesday night, November 2, 1988, Cornell graduate student Robert T. Morris released his worm into the Internet. The worm was intended to spread itself quietly throughout the network by guessing passwords and exploiting bugs in e-mail and other networking utilities. But the program's explosive growth—and the reactions of baffled system administrators trying to protect their systems—virtually shut down the network for two days.

The worm was first introduced into a VAX 11/750 at the MIT Artificial Intelligence Laboratory at about 8 p.m. (EST) on Wednesday night. Within an hour it had spread across the country through the Internet e-mail system. At 6:24 p.m. (PST) it infected a computer at the Rand Corp. in Santa Monica, Calif. At 7:04 p.m. (PST), it infected a major network gateway at the University of California at Berkeley. Fifty minutes later, the worm reached machines at the University of Maryland and Cornell University, and less than an hour after that it had attacked virtually every susceptible machine on the Internet.

The worm also made multiple copies of itself in every computer it infected—so many copies that eventually work on each infected computer ground to a halt, as the machines spent most of their time executing copies of the worm program.

At the University of Utah, the VAX 8600 that serves as the central machine for the computer science department was infected at 9:49 p.m. (MST). By 10:06 p.m. the multiplying worms had rendered the system completely unusable. At 10:20, system programmer Jeff Forys cleared the system of worms; by 10:41 it had become unusable again. At 10:49, Forys shut down the computer entirely, then restarted it. At 11:21, the computer was once again useless because of re-infestation. In spite of continuous efforts to kill the worms, nothing—including bringing down the entire system—seemed to stop them.

Because the Internet connects university and private researchers with military and other government computers, the worm quickly spread to sites where highly classified work is done, including Lawrence Livermore National Laboratories, NASA's Ames Research Center, and the Army Ballistic Research Laboratory at Aberdeen Proving Ground, Md.

At the Ballistic Research Laboratory, staffers initially assumed that the worm was an attack on its systems by foreign intelligence agents. Michael Muuss, who leads the Aberdeen computer systems team, later testified: "We have a history of foreign intelligence activity on our systems, and foreign nations take a great deal of interest in the activities of our research scientists." The weapons lab disconnected itself from the network for six days to make certain that no data had been destroyed or modified.

As it became clear that the worms were using the Internet to spread, some sites "quarantined" their systems, cutting off all connection to the network. The result: though those systems may have been protected, the quarantine attempts crippled the ability of the network to send e-mail—including messages about how to combat the worms.

In the early hours of the following morning, as teams across the country worked on solutions to the worm infestation, an anonymous e-mail message was sent through the network. The message came from Andrew Sudduth, a systems manager at Harvard University's Aiken Computation Lab and a friend of the worm's author; it included instructions for preventing further spread of the worm. But though the e-mail message was addressed to virtually all systems on the network, it was blocked for almost two days by a quarantined computer. Other messages from teams combatting the worm were also delayed by the crippled mail system.

Although the worm was isolated and the network cleared within days, the collapse of the Internet stunned users and system administrators alike. It was the most graphic in a recent string of incidents demonstrating how vulnerable UNIX systems are to software sabotage.

In all, as many as 3000 computers were temporarily disabled by the worm, at locations ranging from universities to military research labs to the National Cancer Institute. The *New York Times* reported that at Carnegie-Mellon University in Pittsburgh, 80 of 100 computers were affected; at the University of Wisconsin, 200 of 300 machines were hit. Though no data was destroyed, one industry association estimated total damage in lost work at nearly $100 million. More realistic estimates place the cost as high as $10 million.

—*EH*

tightly controlled. But he admits that for his company, this is new ground: "We're traditionally a large IBM shop, and I mean large. But UNIX is beginning to come in a lot of places," he says, and without the built-in security that IBM provides, protecting the network is a challenge.

Open Systems, Open Doors?

Is your system safe? Or can you only protect it from worms and viruses by disconnecting it from all networks and phone lines, and carefully guarding it against all outsiders?

As Dennis Ritchie pointed out more than 10 years ago: "UNIX was not developed with security, in any realistic sense, in mind; this fact alone guarantees a vast number of holes." However, there are steps you can take to protect against the most common sources of invasion (see the "UNIX Self-Defense" sidebar).

And although security shouldn't become a full-time preoccupation for system administrators, in the year since the Internet worm it has become a major concern for almost every networked computer system—and that means nearly every UNIX system. Robert T. Morris has been caught and convicted—but the holes and bugs the worm highlighted may be the quarry of a hunt that will last for years to come. □

Features Writer Frank Hayes has covered the computer industry for seven years.

Resources

The CERT Coordination Center hotline, staffed 24 hours a day, handles reports of network break-ins. The hotline's number is 412-268-7090.

Russell Brand's primer on computer security at network sites, which covers topics ranging from prevention to dealing with the aftermath of a break-in, is available free in electronic form or for $10 in printed form in the United States. Contact Russell Brand, 1862 Euclid Ave., Berkeley, CA 94709.

Proper assignment of responsibility for data security

Robert H Courtney, Jr suggests that the primary responsibility for data security should rest with the user community

Reprinted, with permission, from the February 1988 issue, Volume 7 #1
of Computers & Security. Elsevier Advanced Technology, Mayfield House, 256 Banbury Road,
Oxford OX2 7DH

An analysis of the data security responsibilities within an organization is presented. It is proposed that DP management should not have total responsibility, but that this should be shared by staff in the functional areas to ensure cost-effectiveness and viability. By assessing organization structure and data integrity, a clearer picture emerges of the roles and responsibilities of individual staff members.

Keywords: data security, cost-effectiveness, data integrity, organization structure

Data processing management should not have primary responsibility for data security. Although adequate data security often depends, at least in part, on controls which must be implemented and maintained by the DP staff, identification of the need for most security controls and their cost-justification should be the responsibility of other people who are in better positions to do them than are the DP staff. These assertions are derived from our frequent opportunities to observe in operation each of several pertinent factors.

- Security is a people problem. Most of the losses attributable to computer-related data security problems are contributed by people in the functional areas supported by the DP organization and who are not on the DP staff. These people are best controlled by their immediate management and not by the data processing organization.
- Most DP directors have little opportunity for first-hand knowledge of the real consequences to the organization of accidental or intentional modification, disclosure, destruction, or delay of the data on which the functional areas of the business are dependent. Thus, the DP management is in a much poorer position than the functional area management to assess the cost-benefit relationships needed to justify the implementation of appropriate business controls, including data security measures.
- DP directors are rarely measured in terms of the adequacy of their data security program. If they have primary responsibility for data security, they will be blamed for most detected security lapses even though they will usually have had little or no ability to detect or avoid them. They will not be appreciated for their success in providing security. Successes in security are nonevents. Security failures are always more readily recognized than are successes.

The greatest single barrier to the achievement of a cost-effective computer security program is the improper placement of responsibility for data security. The assignment of that responsibility to the DP area which does not have management control over the people causing the problem, which is rarely in a position to recognize flawed data as an indicator of a security problem, and which is in a poor position to assess the consequences to the organization of security lapses is usually a poor choice indeed.

The data processing management must have primary responsibility for the protection for the resources under their direct control. For example, only the DP manage-

River Road, Port Ewen, New York 12466, USA

ment is in a position to lead the development of contingency plans for centralized data processing facilities. Because they must take the lead in contingency planning and in assuring the physical safety of the facilities under their immediate control, it does not follow that there is supporting rationale for the assignment to the DP area of responsibility for all data security.

The managers of the functional areas supported by data processing should have primary responsibility for the security of their respective data aggregations. They should identify needed controls with the help of those with competence in specific security areas, such as the physical security specialists, the buildings and ground staff, and the DP management, but they should retain responsibility for the security of the data on which their functions depend.

The user manager should provide the cost-justification for the controls they need. Where at all possible, particular security needs should be charged back to the respective areas generating those costs. The high security costs of some areas should not be a burden to all users. Distribution of those costs to all users without regard to the specific security needs of the respective areas often leads to a somewhat exaggerated specification of those needs by some users.

If users must specify and pay for security controls, there is reasonable hope of achieving balance in the overall approach to data security. If they ask for too much security, it costs too much. If they ask for too little, they risk losses attributable to them as a result of not having defined their security needs adequately. Further, the internal audit function should see that the control requirements have been defined by the users with adequate rigor and without neglect of important security considerations.

PROBLEM LOCI IN RELATION TO THE ORGANIZATION CHART

Figure 1 depicts the relative sizes of the major computer security problem categories[1]. It is apparent that losses due to mistakes greatly exceed the losses attributable to the other causes. Further, the two largest problem categories are mistakes and dishonest employees* which have these important characteristics in common:

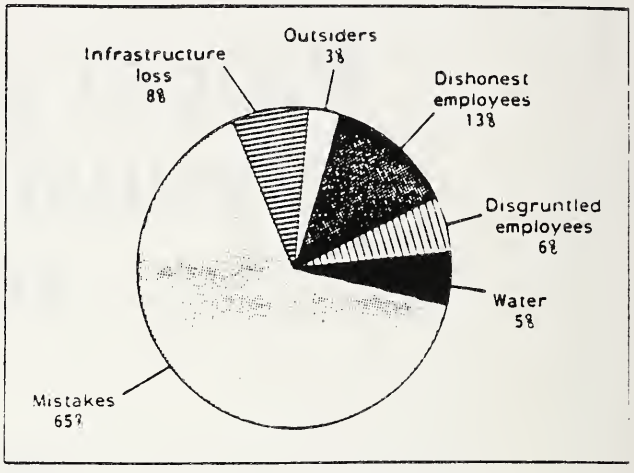


Figure 1. Computer security problem ranking as percentage of all DP security losses

- It is often difficult to tell whether something improper was done accidentally or intentionally, that is, into which of these two categories a particular problem should be placed.
- The most important controls are often equally applicable to both categories. Controls directed specifically at one of these two categories almost always have a proportionately equal effect on the other. This interrelationship forms the basis for our conviction that the crooks will never win over the incompetents in any competition to see who will cause the greatest damage to data.
- The people contributing the greatest portion of the losses attributable to these problems are rarely in technical roles but most commonly, work in the user areas of the organization.
- Problems in both of these categories are frequently neglected in favour of concern for externally originating, more intellectually challenging and potentially less embarrassing problem sources.

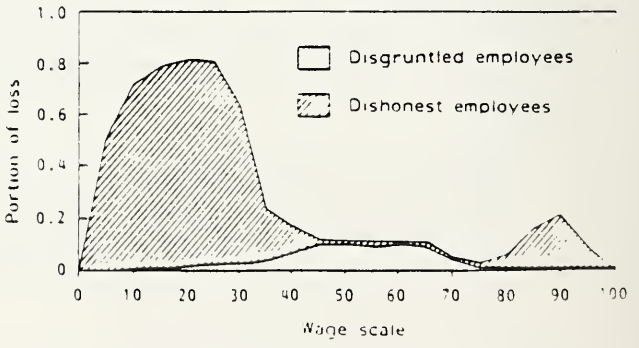


Figure 2. Computer-related employee theft as function of wage levels

*While it is fairly difficult to gather data on computer-related crime, it is almost impossible to compile data on losses due to mistakes through compilations and analyses of incident reports. Our data, which indicate between fifty and eighty percent of the security losses attributable to mistakes, result from a survey of 2 500 organizations which were requested to rank the loss categories and provide gross relative quantifications for them. We have been more successful in assessing the effect of controls on mistakes than we have been in measuring the total losses attributable to them. Nevertheless, we are quite confident that our 50–80% spread is great enough to contain the proper apportionment of losses to that category for most organizations.

Figure 2 relates computer-supported theft by employees to their respective salary levels. As the figure shows, slightly more than 80% of the theft by computer was contributed by employees in the fourth and fifth salary quintiles, i.e. in the bottom 40% of the wage scale. The great majority of these people are clerical employees in data entry, file maintenance, and query responses while most of the others are lower level administrative and operational employees.

Losses to persons in the first salary quintile, the most rapidly growing group, are in second place*. Even though the losses in this employee category are growing rapidly, now, we estimate, at about 18% per year, this growth rate will not be maintained and those losses will not approach, except in some very unusual and probably localized circumstances, the losses to the employees in the lower 40% of the salary range**.

The people best equipped to mount the more sophisticated intrusions into the data processing complexes, who are most often described in the general and trade press as posing a major security threat because of their technical strength, report to the data processing management and are usually in the second and third salary quintiles. Contrary to conventional wisdom, these people are responsible for only a very small portion of employee theft through misuse of the data processing resources. To the extent that we have security problems presented by the technically strong employees, they usually do not involve theft but more commonly reflect a desire to do harm to the organization as a result of morale problems of one type or another. There is a striking correlation between the amount of damage done by technically oriented data processing people and the imminence of such potentially disruptive events as mergers, acquisitions, and divestitures, which threaten the stability of the current DP organization. This aspect of computer security, unlike the overall data security problem, is best controlled by the DP management to whom those people report.

The people contributing the major losses shown in our figure are, for the most part, not on the DP staff; they work in the DP user community. No control which limits the resources to which they have access will be very effective in controlling their misconduct because they abuse the resources to which they must have access to get their assigned work done. On the other hand, the problems that they contribute can usually be managed into submission by appropriately chosen combinations of personal identification, logging, and a carefully planned and intelligent processing of those logs.

*These data came from the histories of 1 474 cases of computer-related theft or malicious damage in the three years ending August 31, 1987. We cannot know what portion of the total number of such cases these represent, but we do believe that they constitute a statistically significant sample. Although we have data gathered over 11 years, we do not include those earlier data because they are not descriptive of today's operating environment.

**For the first several years we gathered data on this subject and observed a very low theft rate by people in the highest salary quintile. We naively attributed this to the greater integrity of those more senior people. We did not give adequate consideration to their lack of technical competence in our gathering of data on computer-related crime. That situation is now changing rapidly as the number of on-line PCs in the upper echelon offices continues to increase.

ASSESSING DATA INTEGRITY

The usefulness or relative value of data are usually heavily dependent upon the integrity of those data. But the accuracy of this statement is, in turn, wholly dependent upon the acceptance of a definition of integrity which does not imply perfection. Integrity should not be used to mean completely free of flaws, a condition which is usually unachievable in any realistic, cost-effective DP environment.

Whether we are discussing data, people, systems or programs, we have found it convenient to define integrity as 'the property of being no worse than thought to be'. A system does not have to be perfect to have integrity; it need only be no worse than we think it is, i.e. free of unpleasant surprises. We can often do good work with some rather poor people, but only if we know just how poor they are. Similarly, we often work quite comfortably with data that are far from perfect provided only that we know what the limitations are on the accuracy, completeness, timeliness, and privacy of those data. It is when the data are worse than we think, that is, when their integrity is impaired, that we most commonly encounter trouble using them.

If we need to know the integrity of our data for them to be fully useful, who is to have the role of assessing that integrity, of specifying the degree of degradation that can be tolerated, and finally, of specifying the controls necessary to the realization of the needed integrity goals? The very important answer, and the one which is fundamental to the notions presented here, is that only the management of those functional areas using those data are ordinarily in a position to make those determinations, including assessments of the cost-benefit relationships between losses and the cost of the controls necessary to the realization of the required integrity.

The information about the data integrity requirements of each user area can be developed, documented and presented to others who might then be responsible for data security, but this will not work well. It dilutes the responsibility for security. It also virtually assures some intentional warping of the data presented to the security staff to induce them to do what the users want them to do.

The data processing staff are rarely in a position to evaluate the quality of data, to determine what the quality is and what it needs to be, to see whether specific fields are correct, whether they have been neglected, or whether they have been manipulated in

support of an attempted theft. Thus, it is inappropriate that the DP staff be put in the position of defining the controls through which such problems can be identified and minimized.

WORKABLE ASSIGNMENT OF RESPONSIBILITIES FOR COMPUTER SECURITY

In the table immediately below are listed some of the more important tasks necessary to a reasonably complete computer security program. For each we have provided an indicator of which group, the DP staff or the users, should be in the best position to do the work described. We recognize that the user community in any particular company may today be totally unprepared to do many of the things we have said are best done by them, but that condition can be, and in several more progressive companies has been, rectified.

Security factor	*Best done by* User areas	DP staff
1. Estimating the cost of technical security measures, such as logging, access control, back-up of secondary storage, etc		X
2. Contingency planning		
(a) Provision of data on consequences of disruption	X	
(b) Preparation of plan		X
3. Data integrity requirements	X	
4. Value of controls in terms of displaceable loss	X	
5. Source of information and general guidance in all technical aspects of computer security. Assistance to users in understanding technical security exposures		X
6. Human factors aspects, including need for and probable effectiveness of controls for holding individuals accountable for each record entered, modified, or read	X	
7. Need for data classification program (in conjunction with Legal Department)	X	
8. Devising a data classification program (in conjunction with Legal Department)		X
9. Evaluation of data to detect errors and intentional misconduct	X	
10. Controlling security-related misconduct by user-area employees	X	
11. Controlling security-related misconduct by DP staff		X
12. Preventing damage from technical intrusions by outsiders, such as through in-dial ports, wiretapping, etc. as may be cost-justified by users		X
13. Installing, maintaining, and administering access control programs, such as RACF, ACF2, TOP SECRET, DB Secure, and Guardian		X
14. Determination of who is to access what data	X	
15. Integrity of software developed in-house or which they have been asked to evaluate		X

Primary responsibility for the security of data should be placed, by corporate policy, with the respective functional areas which are the principal users of each data aggregation. As we have noted above, they are in the best position to assess the importance of those data to the whole enterprise, to assess the consequences of any loss of data integrity, to control the people who have the greatest influence on data integrity, and who are responsible to the corporate management for the successful conduct of the functions supported by those data.

Secondary responsibility should be placed on those to whom access to data is given by those with primary responsibility. Secondary responsibility entails

mandatory compliance with the security guidelines established by those with primary responsibility for those particular data.

The data processing organization should have only custodial responsibility for the data which are under their direct control so that they can provide safeguards agreed upon with those having primary responsibility. In addition, the data processing organization should establish a Computer Security Competence Center with a size appropriate to the needs of the particular organization. This group should have the following roles and responsibilities:

- Provide advice and consultation to the user community on computer security matters.
- Maintain awareness of the state-of-the-art in computer security through professional organizations and peer contacts so as to bring early awareness to the DP staff and the users of significant developments in both security measures and evolving problem categories.
- Administer the access control package(s) which run on facilities under control of the DP organization. In this role, the administrator will only permit access to those persons authorized access by those with primary responsibility for data security.
- Take leadership responsibility for determining, by working with the user areas and the Legal Department, the need for a data classification programme. If one is needed, take the lead in developing one that satisfies the potentially diverse needs of the user community.
- Develop an Employee Data Security Awareness Programme which will adequately sensitize employees to the importance of their cooperation in the corporate data security program.
- Develop educational programmes for user-area management in the use of currently available audit packages for the critical review and evaluation of their data. These users need to be able, whenever possible, to detect data integrity problems before they become any more costly than necessary. This matter is complex and quite beyond the scope of this particular paper — but it is nevertheless very important.
- Extract from the user community sufficient information about the dependence of each on the continued availability of their data and of the means of processing them and about the economic consequences to the enterprise of their loss forming a basis for the formulation of a contingency plan. Again, a complete discussion of contingency planning is beyond the scope of this paper, but it should be made clear that responsibility for the development of contingency plans for facilities under control of the data processing organization should be with this group.
- Develop an educational programme which provides guidelines in PC security, including back-up and recovery, for the whole enterprise.

CONCLUSION

Every experienced reader will realize that what has worked well for others will not necessarily work well in his particular situation. The plea here is not for complete emulation of what others have done successfully in establishing responsibility for data security, but only recognition of the factors which influenced those who have been successful in establishing workable, relatively unobtrusive and yet credible computer security arrangements in their organizations.

Experience in consulting on computer security with a few hundred companies in a quite diverse array of industry areas leaves one completely convinced that the primary responsibility for data security must, in almost all companies, be with the user community if it is to be realistic, adequately comprehensive, and cost-effective. There is an important computer security role indeed for the data processing organization, but that role should not include seeking or accepting primary responsibility for the security of the users' data.

REFERENCES

1 Courtney, R H and Todd, M A 'Problem definition: an essential prerequisite to the implementation of security measures' IFIP SEC '84, Toronto (1984)

Assessing Security

WHEN IS A LAN AT RISK, AND HOW MUCH CAN A COMPANY LOSE?

by Peter Stephenson

Every network needs security, but each at a different level. How much security depends on your level of risk and exposure. But security need not be a costly thing. Seventy-five percent of protection is administrative, not technical. Assess your risks, evaluate your exposure, design a security plan, and then enforce it.

Hiring a security consultant *before* you need to rescue your data from a virus can be far less costly than hiring that consultant after the virus has done its dirty work. And, if you call the right specialist, you may hear some things about security that will surprise you. For example, the consultant might tell you that 75 percent of the protection you need is administrative, not technical. You may also find out that a reasonable investment in a streaming tape backup system could avoid serious loss. The most important question the consulant will ask you is: Why do you think you need more security on your system?

Exposure Assessment

Security frequently is compared to locking a door in your house. Certainly that is an apt comparison in some cases. However, I lived in a small town in Michigan for years and never locked my house. I can't recall a robbery in that rural community during the time I lived there. But when I moved to the Detroit area, it was a far different story. Everything gets locked. Twice.

The level of security I needed was based on risk and exposure. In the small midwestern town, my risk was not great. Everyone knew everyone and crime was not an issue. The risk went up when I moved to Detroit. In the small community, my exposure was limited; I had one computer and it wasn't worth the trouble to steal. Also, I lived in an apartment. A burglar would have had to break into my home by passing several other apartments, some of which had snoopy neighbors watching every move in the area. By the time I moved to a house in the city, I had a lot more equipment to lose and fewer neighbors to keep watch. So I was not only at greater risk, I was more exposed.

Security begins with an honest assessment of what you can afford to lose and how likely you are to lose it. Let's start with what you can afford to lose. Government agencies have estimated that it can cost $2,000 per MB of lost data to replace data after an intrusion or accident (accidents cost users a lot more data than intruders do). But if your data is trivial and well backed up, is it really necessary to go to great expense to protect it? Data loss could be limited to one work day if you back up properly.

Furthermore, if data resides on a LAN of fewer than 20 workstations with no connection to the outside world, how great is your exposure? The truth is that your risk and exposure are likely to be greater due to careless users than determined crackers.

The first question to ask yourself when assessing your risk is: At what point is it more expensive to reenter lost data than it is to prevent the loss? If you can recreate a week's worth of data entry in a half hour, it probably doesn't make a lot of sense to spend thousands of dollars on sophisticated security and backup equipment. But if you have very heavy data entry, loss of a half day's input could be disastrous. Almost any reasonable cost to prevent loss would be justified.

How about exposure? The box lists questions to help you determine how exposed your data is to loss. Other exposure factors not listed might be peculiar to your installation. Physical office layout can have a big impact. Not only does do intruders play a part in exposure, but also the cleaning crew—it could hit the server power cord with a vacuum cleaner and crash your LAN while the night crew is entering data.

If you get the idea that a big part of risk and exposure assessment is a combination of common sense and heading Murphy off at the pass, you're very close to the target. It's a fact of life that more data is lost and damaged through carelessness than through planned intrusion.

LAN Babysitting 101

As the network supervisor, you have 100 percent of the responsibility for keeping your data secure. Once you formulate a realistic assessment of your risk and exposure, you need to put together a security plan. Remember that a part of any risk and exposure assessment is political. The issue may be less what *you* feel your company can afford to lose and more what your boss feels the company can lose. Management never wants to lose any data. You know you can afford a certain level of loss before it begins to make sense to spend money to prevent it. Err on the side of conservatism.

While you may be correct in your assessment of normal conditions, conditions are rarely normal. Workers get sick, power goes funny, equipment fails, all at the time when you have to replace lost data. Give yourself a bit of slack as you plan for security. It's a law of nature that you'll need it.

When you've built your security plan, write it down. Every network with more than 10 users should have a written security manual. Two reasons are behind this necessary step. First, with a security manual, you do away with any questions about how the network is to be run and how to recover from a disaster. Second, creating such a manual is a step toward taking network administration more seriously. Because most data loss is accidental, it follows that accidents can be mitigated, if not prevented, when users have a healthy attitude toward LAN security.

Now the hard part starts: enforce-

ment. That's a very strong word for a simple, nonthreatening concept that boils down to positive attitude reenforcement. The network administrator is, in most cases, a babysitter. He or she spends a lot of time teaching, handholding, and inconspicuously watching network users. Developing positive attitudes about safe computing is one of the most important parts of the LAN supervisor's job. When the administrator fails to do that part of the job well, the rest of the job is far more difficult.

After Groundwork, Action

Now that we've set the stage, let's address the threats, risks, and exposures with guidelines to keep your network safe. A rule of thumb is never risk more than you can afford to lose. Now we'll return to our exposure list and answer some of those questions the way you should be able to if your LAN is secure. By secure I mean within the context of your level of acceptable risk and exposure. There are no (or, at least, very few) absolutes when it comes to answering the questions.

How well trained are your users?

Training is key to good network administration and safe computing. An unbelievable amount of data is lost or corrupted because inexperienced users made errors that could have been prevented by training. Often prevention comes from simply knowing what they can and can't do on the network. Every new user should read the company security manual. The manual should be reread periodically as a refresher.

Often the ability to avoid data loss starts with the recognition that something has gone wrong. Users are your best observers of network performance. They need to be trained so that they can be network assets, not liabilities.

"Magic key" sequences should be avoided unless a solid menu system is in place on your LAN. Users should understand their computing environment, not just the keystrokes to do a job.

How many users do you have?

The larger the network, the greater the exposure. When LANs grow larger, it is a good idea to have workgroup administrators. With many users, you need greater security and more stringent controls on how the network is used.

Is your data organized on a need-to-know basis?

This may be the most important question affecting exposure. The best security against common accidents and errors is to keep users out of data they have no reason to use. This is a two-level task. First, it applies to data segregation based on the employee's right to the information. Second, it implies that novices should be limited to the areas in which they are thoroughly trained.

The importance of keeping users restricted to the data they both need and know how to use can't be overemphasized. Management consultants evaluating network administration often find large amounts of data opened to corruption by naive users who have no need to get anywhere near that information. Also, when data and programs are available to all users on a large network, deliberate damage by a disgruntled employee becomes greater.

Do you interconnect with other networks?

From a technical standpoint, this has the greatest potential for introducing damage from intentional or unintentional virus attack. The big difficulty is that usually there's no way of tracking down a virus or knowing that it's being introduced until it's too late. If you interconnect with other networks automatically (polling schemes and the like), you need to be careful of where executable programs go when they're introduced into the system. And you need to keep a detailed audit trail of all internetwork connections. Know to whom you've connecting, when you connected, and what happened during the connection.

Also, executable files (not data) must be isolated after they are transferred to your network until you can check them for contamination or damage. A damaged program can unpredictably hang your network. File transfers of archived files can often produce unpredictable results when you want to unpack the file.

How high is user turnover?

When user turnover is high, additional levels of isolation for those users in sensitive areas is a must. This is one of those rare situations that demands periodic password changes.

I have some strong opinions about

password changes, by the way. Conventional wisdom says that passwords should never be shared and should be changed frequently, often by the system itself. I don't completely agree with conventional wisdom: Humans react negatively to frequent password changes. People tend to move their data to the hard disk (if they have one) on their own PCs to avoid using the LAN if security gets in the way of utility. This, of course, is what you don't want to happen.

So if you have a high-user turnover, isolate workgroups on the LAN. Virtually all network operating systems allow the formation of groups. For sensitive groups assign a two-level password scheme. The first level is for the group. The second is for individuals. Change the group password whenever you suspect a compromise or have a turnover in the group. Let the users change their passwords when they wish. Keep the group passwords simple. Discourage putting user passwords on boot disks, but allow the group password to be automated if you feel that you have adequate security of the boot disks.

Do you have dial-in or dial-out phone lines?

This is a variation on internetworking and carries the same warnings, with one exception. If dial-out lines are accessible universally, you stand a serious risk of abuse. If every user can easily dial out on a modem, there will be a tendency to connect to Bulletin Board Systems (BBS) and other time-wasting, potentially dangerous sources of data and files. You do not want uncontrolled downloads of files from BBSs, which are a primary source of virus infection. Also, many BBS systems are toll calls.

What's the solution when you have a legitimate need for dial out? Start by using a communications server. A communications server is a separate machine on the LAN that allows network users to connect to the outside world through one modem. To make this more attractive, you might consider a combination facsimile and modem board; you have the advantage of making both dial out and fax available to all network users.

But here's the catch. Since the communications server's use can be restricted, you can do two things. First, you can restrict who uses it and when. Second, you can keep a detailed log of all communications server activity. With these two controls, those who need outside connection can have it, under your scrutiny, and you will know what they did use an outside connection and when.

Are dial-out lines accessible to all users?

Following up on the preceding question, the answer to this one is usually no. All users rarely need outside connections. When they do, however, use the communications server approach and write scripts that automate (and restrict) the outside numbers to which users can connect.

Do you have a procedure for screening new software before it is placed into service on your LAN?

Never allow a new program, regardless of where it came from, on your network until you have screened it. Screening has several levels. For a commercial program delivered from a legitimate supplier in its shrinkwrap, screening may consist of checking the package for damage and installing it. For a utility brought in by an employee, screening may include a virus scan on an isolated PC.

Never load an executable program that isn't delivered directly from a commercial supplier. Don't allow disk swapping; it's piracy, illegal, and dangerous. Discourage personal programs on your LAN, but if you do allow it, screen them for damage or viruses. Do your screening on an offline, isolated (standalone) PC. Include such procedures as scanning with antivirus programs, moving the system date forward to dates such as April 1, Friday the 13th, or Columbus Day, and checking files on the isolated machine for changes in length.

How many network administrators do you have?

On large LANs, you may need multiple network administrators. Network administrators can have a lot of power. And, if they are disgruntled, they can do a lot of damage. If you must use several administrators, grant privileges selectively; that is, avoid global privileges for all but the main network administrator. Give local administrators the privileges they need within their work or responsibility areas. Stick to the need-to-know rule with administrators and users and you won't go

wrong.

How do your users boot their machines?

There are many ways to boot machines on a LAN. Putting the boot files on the hard disk is convenient, but it's also a good way to invite unauthorized use. So, unless your LAN is very small, don't give in to temptation. Use floppy boot disks and keep them secure when not in use. How secure goes back to the risk and exposure assessment discussed earlier.

What hardware comprises a typical workstation?

The ideal workstation, for security and utility, is the diskless workstation. They can't be force-booted by an inside hacker, they tend to have more useable application memory, and they are usually nice, easy-to-use color monitors. They also cost more than many PCs and most networks grow out of a collection of single-user computers, which are converted for LAN use.

How can you have your cake and eat it too? If you determine that you are at risk and highly exposed, make it a rule to replace old, dying PCs with diskless workstations. When you need a new workstation, make it diskless. If that's overkill for your situation, limit workstations with hard drives to those users who absolutely must be offline from time to time, or to server users on a peer-to-peer LAN with nondedicated servers.

What is the typical state of user morale?

Low morale demands high security. After accidents, the most common source of damaged, compromised, or lost data is the disgruntled employee. These folks are very dangerous because they are, in security terms, trusted users. They also are often hard to ferret out since they rarely signal their intentions to intrude company data. They also know how to use the LAN.

What applications do you run?

Database applications are the most vulnerable to intrusion. They are also the most likely to sustain severe loss if compromised. However, industrial strength databases often have audit trails and individual passwords. Make constant use of audit trails, and use nested passwords in accordance with your risk and exposure assessment. Remember, people won't use applications that are too confusing or time consuming to access. So use balance to protect your data without discouraging authorized access.

Do you connect to mainframes or minicomputers?

This is similar to the dial-out line problem, but not as tough to handle. These connections are, usually, dedicated and controlled at the mainframe end. A good bet, though, is to include the connections on the communications server described earlier. If you confine all outside communications to a communications server or appropriate bridge, you'll control access without getting in the user's way.

What is your backup procedure?

Volumes could be written on how to back up your LAN. But it all boils down to two pieces of advice: do it and use a grandfather system. You should back up as frequently as your situation demands. That could be several times a day in extreme cases, but it should be at least daily.

By grandfathering, you use a schedule that insures that you won't lose important data, even if you are hit by a virus. First, make a baseline backup so every file on the LAN is backed up. It is the basis you'll use if you must restore the entire system. Second, make daily backups, which you keep until the end of the week. At week's end, you do the weekly back up, which you keep for a month. Return the daily tapes to use for next week's backups.

Now, each month, do a monthly backup, which you keep for a year. Thus, you have four dailies. On Friday, you keep a weekly. At the end of the month, you have four weeklies. At the end of the year, you have 12 monthlies. And you always have the baseline backup. Only back up files that change (incremental backup). Then when you need to restore an entire drive, you start with the baseline and then update with the most recent incremental backup. If you find a virus in your backup, you can move to an earlier backup (that's why you grandfather) and the worst you'll have to do is reenter a month's worth of data.

Do you have a disaster recovery plan?

It's a subject in itself, but the bottom line is, you'd better. If you don't have a plan, you'll lose data. It's not *if*, it's *when*.

How do your users view network security?

This is a tough one. If you use too much security, your users may avoid it. You don't want that. Part of the reason for a LAN is that you can control the data your company uses to keep it pure and up-to-date.

The solution is to use as much security as your risk and exposure assessment demands. Educate your users about the level of security you deem appropriate. Keep education positive and upbeat and, if possible, don't provide a place for your users to offload data and avoid LAN use altogether.

How complicated is your current security scheme?

Continuing in the same vein, the least complicated security is used consistently. But don't be tempted to automate the login process unless your boot disks are well secured. It's never a good idea to include passwords on boot disks, though. If you must automate, leave out that one important aspect. First, it invites intrusion. Second, it complicates later changes to passwords.

How clean and consistent is your power?

This is often overlooked. You can lose lots of data with no intrusion or user accident if you connect to dirty power. If your risk assessment includes dirty power, get an Uninterruptable Power Supply (UPS) for every vulnerable computer. Here again, use your risk and exposure assessment to decide where you need a UPS.

Keeping It Your Business

The bottom line for LAN security is very simple. Decide what you realistically need in security with your risk and exposure assessment and then apply the security measures. Have a disaster recovery plan and a written security document that lays down policies and procedures.

Infuse your users with a sense of security and responsibility for the data on your company's LAN. Isolate users and subadministrators to the data they need to use. Perform regular, grandfathered incremental backups and maintain a detailed audit trail of LAN activity so that if you do have a problem you can track it down more easily. Limit or, at least, control access to outside systems. If you must internetwork, be sure that you maintain audit trail records, again, to track down problems. Finally, screen all new software before you put it on your LAN.

Security is as much about preventing data loss from errors as it is about preventing intrusion. And intrusion, when it occurs, is likely from someone inside the company. You can gain a lot of additional security on your LAN if you use common sense and apply businesslike administrative procedures. Security is meant to keep your business your business. But it's also meant to help users do their jobs safely and conveniently. If you carefully perform risk and exposure assessment, you will have a system that can work for your company and give you far fewer headaches. ❑

NIST Group Explores Risk-Assessment Packages

Works to understand assets, vulnerabilities, threats and costs of safeguards

By GARY H. ANTHES

Security specialists from the National Institute of Standards and Technology and the National Computer Security Center (NCSC) have embarked on a project to dream up disaster scenarios – the worst nightmares of computer managers, including computer center fires, destruction of data, viruses and other mischief.

Anticipating the worst is all in a day's work for the people in NIST's Risk Management Research Laboratory, established 18 months ago to advance the state of the art in assessing and managing the risks associated with maintaining and using computer systems.

Putting together the risk scenarios is the second phase in a major effort by the lab to improve, expand and standardize the patch-quilt of methods that have evolved for identifying risks and for safeguarding against them. In the first phase, nearing completion, the lab evaluated two dozen risk-assessment packages and used them as the basis for constructing a conceptual framework for risk management.

The laboratory this year is funded mostly by NCSC – a unit of the Defense Department's National Security Agency – and staffed by personnel from NIST. The laboratory was established in part as a response to an Office of Management and Budget directive that in 1985 mandated periodic risk analysis for all federal systems.

It also was formed as it became apparent that the need for risk management was increasing faster than the sophistication of the tools available to support it. "Everyone says do risk analysis, but no one really knows how to do it," said Eugene F. Troy, head of the Computer Security Assistance Group of NIST's Computer Security Division.

The software packages examined so far all reflect different ideas about what risk assessment means, he said. "Risk analysis can run the gamut from back of the envelope to a quarter-million-dollar report."

Basic Framework

Nevertheless, the packages have provided a basis for the framework, which Troy said will give guidance to agencies that wish to improve the security of their computer systems and facilities. The framework also can serve as a checklist for preparing requests for proposals and for evaluating vendor offerings, he said.

The objective of the framework is to tie together in a systematic way key elements involved in risk management and to explain their relationships qualitatively. The elements include assets, threats, vulnerabilities, consequences, safeguards and the cost of safeguards.

The lab also will present for comment drafts of its risk scenarios.

Each of the packages examined so far addresses part of the risk-management process, but none is all-encompassing, said Irene Gilbert, who heads the lab. Another problem is that packages alleged to measure the same things do not always give consistent results.

The standard test scenarios – which will encompass areas such as applications software, computer networks and data center facilities – can be used to evaluate and calibrate the packages. They can be used to see if a particular package or approach is able to identify the security flaws built into the test cases, Gilbert said.

The lab will not develop its own packages, nor will it test or endorse them, Gilbert said. The framework and scenarios may be used by agencies and vendors to develop new risk-management methodologies, Gilbert said. "We want to encourage vendors to develop packages that meet the requirements of a broad spectrum of computer environments."

Troy said that developing risk-management packages is not a trivial undertaking, with companies typically investing more than $1 million in them.

Work on the framework and the test cases eventually will find its way into a new Federal Information Processing Standard on risk management in federal computer environments. The FIPS is likely to be published in 1991 or 1992, Troy said. ◄

Crackdown on software pirates

Industry watchdogs renew efforts to curb illegal copying

BY JANET MASON

Software piracy wears a reputation of cloaked intrigue — Dick Tracy tracking down illicit retail operations on crowded Hong Kong streets and corporate bad guys churning out pirated disks for companywide use.

Less notice has been paid to the casual pirate — the well-meaning employee who replaces the company-issued word processing program with his own and then allows co-workers to copy it. Or the staff member who pirates company disks for home use.

But more and more, both types of pirates are finding that they are risking more than just a guilty conscience by illegally copying software. Corporations, computer dealers, rental operations, universities and bulletin boards are increasingly being taken to task by software industry watchdogs.

In recent months, industry trade associations have begun stepping up educational and prevention efforts as part of an anti-piracy crusade. Groups such as Adapso, a Washington, D.C.-based computer and software services association, Software Publishers Association (SPA) and Business Software Alliance (BSA) have been consulting retailers and the publishing industry on how to avoid copyright infringements.

Moreover, the maturing software industry has begun to initiate the same protective steps taken by other intellectual property industries, such as movie, record and book companies, according to legal experts.

"In the past 20 months, [SPA] has brought 30 lawsuits against offenders," says Mary Jane Saunders, SPA general counsel and an attorney who handles domestic piracy issues (see story page 115).

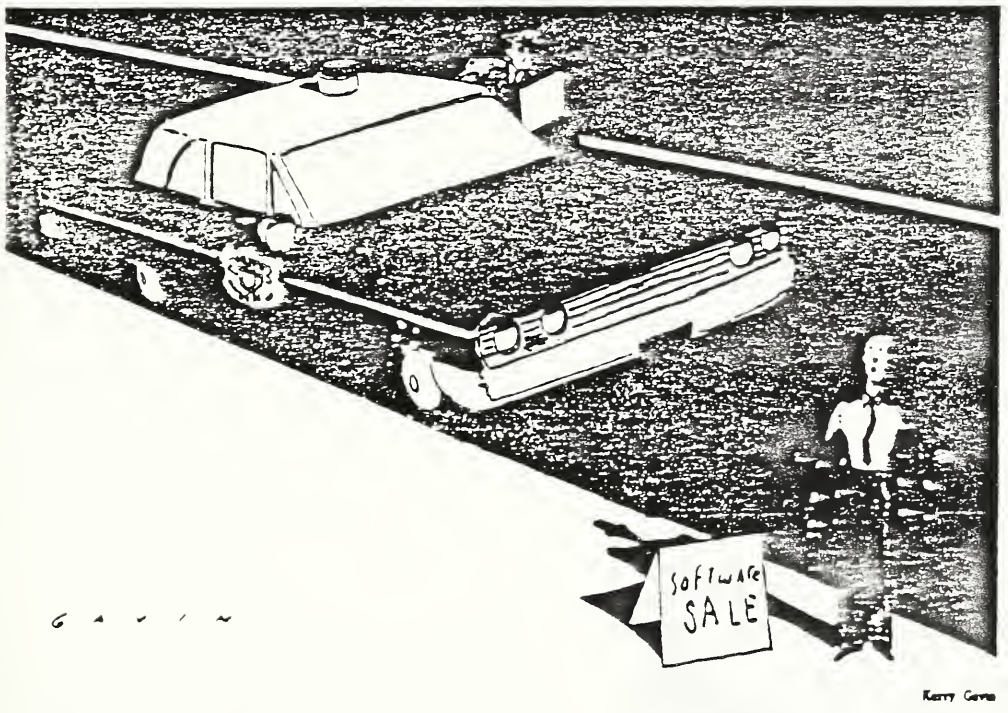
"The SPA has plastered the world with brochures to inform people that piracy destroys the valuable resource of commercial software," says Donn B. Parker, a senior management consultant at Menlo Park, Calif.-based SRI International, Inc.

The general notion of software piracy as an unethical practice is being driven home by expensive lawsuits brought against major corporations and other offenders pirating software.

Although those who make illegal copies are rarely prosecuted, purchasers of large amounts of illegal software can receive a $50,000 fine under U.S. Code Section 17 Copyright Law.

With the help of an amended trade act, international strides have been made in closing the doors of illicit retail and mail-order software operations in the Far East and filing lawsuits against European corporate infringers.

Mason is a Philadelphia-based freelance journalist.



Kerry Gove

Counterfeits abound

Software vendors say they are fighting for their very existence in moving against piracy.

In what he deems a conservative estimate of the amount of U.S. illegal software in circulation, Peter Beruk at SPA says that "for every legal software

- **U.S., foreign offenders eyed**
- **Annual losses estimated at $4.1 billion**
- **'Piracy destroys legal software'**

package in use, there is another illegal one."

Some say that that number is as high as five illegal copies for every legal one, adds Beruk, who is a litigation project manager at the Washington, D.C.-based association.

Adding fuel to the fire are state agencies, which have found — in a precedent-setting court case involving the University of California at Los Angeles — that they are currently exempt from U.S. copyright law. And rental operations, both legitimate and those renting pirated disks, are under increasing scrutiny. Along with personal computer software, which is easily copied, cumbersome mainframe software has also had piracy problems, though more rarely.

SRI's Parker defines software piracy as a "crimoid" in the sense that it is one of a string of hot computer news topics that include invasion of privacy, hackers and computer viruses.

"It became a crimoid in the early 1980s," Parker explains. "At that time, [software piracy] was seriously threatening the computer industry to the extent that it was about to destroy it."

For software vendors, the intrigue of piracy translates into stark bottom-line losses. Worldwide hardware and software losses from copyright infringement total $4.1 billion annually, says Tom Sherman, an analyst at the U.S. International Trade Commission.

Since then, software companies have tried a series of technological solutions, such as, most notably, disks that cannot be copied, which were "all thwarted by an army of hackers," Sherman claims.

Warning shot fired

Corporations received a warning last June when Facts on File, Inc., a New York-based publishing company, reached a six-figure out-of-court settlement with five software vendors that, with the help of SPA, filed a suit on the grounds that the company had copied their programs.

The case was the first to be filed by multiple vendors against a corporation. SPA has also settled four other corporate piracy cases out of court with the provision that the organizations' names would not be released. Currently, the association has six cases pending against corporations.

As a result of the Facts on File case, a major accounting firm posted articles pertaining to the lawsuit on bulletin boards in all of its corporate locations with a memo from top management saying, "Don't let this ever happen to us." SPA's Saunders points out that the same firm encourages its clients to have corporatewide piracy contracts and audits.

SPA has pursued informal corporate software audits with two dozen companies to avoid the expense and embarrassment associated with lawsuits. With the company's cooperation, SPA sends in its own investigator to compare installed software on hard disks with corporate purchasing records.

"When we find pirated software," Saunders explains, "the company has to destroy it, pay us a penalty — which is less than if we had taken them to court — and then buy legal copies of the software."

He says that his organization has found that if a company does not pay attention to software — despite its policies — it is likely to get pirated. SPA usually suggests a corporate audit when it finds a company with an unenforced corporate software policy. The association also provides a self-imposed contract and audit program for companies.

Policy needed

Parker emphasizes that a corporate effort against software piracy starts with an organizational policy, backed up with software audits and followed up by swift punishment of offenders.

The contract, which Parker suggests employees should sign once a year, should state that, "We, as a company, do not engage in software piracy, and our purchasing function should pursue site licenses and larger scale purchasing."

Parker says the contract may help the company in a lawsuit — provided that the policy has been enforced. His recommendations come through SRI's Information Integrity Institute, which acts as a clearinghouse for 50 major companies to share information on computer security, including piracy.

Parker also suggests that companies code their disks by buying corporate disks in a certain color or brand so that

Copylefts vs. copyrights

Mention software piracy to Richard M. Stallman, and you're in for a novel response. "What's that?" he'll ask. It's an unusual reaction for somebody entrenched in the software industry. However, given Stallman's background — programmer, computer industry outlaw and MIT's "last hacker" — his answer is far from surprising.

"Software licensing is antisocial, because it keeps information away from your neighbors," Stallman says, "and it prohibits the growth of the technology."

Stallman refuses to use any licensed commercial software and has spent hours in a cramped MIT laboratory developing programs that enable users to view the source code and improve on it if they wish. Based on the hackers' creed that all information must be free to further collective societal knowledge, he has founded The Free Software Foundation as a legal alternative to copyrighted software.

Housed on MIT's campus, the foundation provides "copylefts" that ensure programs are freely distributed and not incorporated into for-profit programs. Despite its renegade attitude, the foundation is supported by some corporate heavyweights. The Next, Inc. computer comes bundled with the software. Other firms, including Hewlett-Packard Co., Bull H. N. Information Systems, Inc., Nynex Corp. and NCR Corp., support the foundation.

The programs — which run on Digital Equipment Corp. VAXs, other minicomputers and Intel Corp. 80386-based personal computers — are not exactly free. They are sold on magnetic tapes that cost $150 plus shipping and handling. However, as Jay Fenlason, one of the foundation's two full-time programmers, says, "Our primary purpose is to develop software, not to make tapes, so we try to discourage people from buying tapes and instead make copies from their colleagues."

The foundation distributes an entire development system that ultimately will include applications for both programmers and nonprogrammers and an operating system that reportedly rivals Unix.

Currently, the most popular program is an editor called Emacs. The foundation also distributes a widely used compiler, and programmers are working on a spreadsheet product.

Despite the high level of corporate backing — which, along with individual contributions, supports the foundation — not everyone is pleased with what the group stands for.

One commercial software analyst accuses Stallman of creating an operating system designed "to put Unix out of business." Stallman counters this by explaining that he is simply creating a program that people need, which will further society by encouraging the spread of knowledge.

However, there are those who could not disagree more with Stallman's views. Legal threats aside, opponents say that pirating software carries repercussions that can damage the productivity of an entire industry and individual users.

Mary Jane Saunders at Software Publishers Association (SPA) argues that software piracy negatively affects all users by driving up prices and diverting funds used for research and development.

Saunders, who is general counsel at the association, says that about 15% to 20% of corporate profits are returned to R&D, which later benefits customers in the form of software product updates and new versions. "If vendors don't have the revenue to match distribution of their products, they don't have much to return to R&D," she says.

Ultimately, piracy could result in fewer new products introduced to market, she says. "If people are stealing vendors' property, there is no incentive" to develop new packages, she claims.

Immediate repercussions of using pirated software include the lack of documentation and technical support. "In the short term, you get the program," says Peter Beruk, SPA's litigation project manager, "but you won't receive documentation or be able to call the support number without a serial number."

JANET MASON

employees can distinguish between corporate and personal property. "This way," Parker says, "employees are aware that they are engaged in illegal activity with company materials."

Saunders says that "there are people in corporate America who understand copyright law." Many of them are among the 20 callers who contact SPA's piracy hot line every day and are willing to sign an affidavit about piracy in their companies.

People who call the hot line include former employees, consultants and, in one case, a temporary secretary. This secretary, who writes novels in her spare time and, as Saunders observes, "is concerned with protecting her own intellectual property," reported three piracy incidents at separate companies.

Dealing with dealers

Along with tracking corporate abuses, SPA also investigates calls about computer dealers and bulletin boards. In most cases, SPA tries to settle out of court on behalf of the 565 software vendors it represents. However, Saunders explains, there are many cases in which litigation cannot be avoided. In a current case, SPA has been forced to take a California computer dealer to court even though the dealer contends that "the only time it ever loaded a hard disk with free software was when our [SPA] investigator was there."

SPA has filed suits against 10 computer dealers, and its efforts and corporate awareness programs may be paying off with fewer lawsuits.

Recently, an SPA investigator checking a report of piracy in a Florida dealership found no evidence of piracy in two dozen other area stores. "A year and a half ago," Saunders says, "he would have found at least half of them doing something illegal. This time, he even found our brochures in some of the stores."

As for bulletin boards, although the association has pressed charges against five systems operators, the group tends to warn those involved rather than filing suit. "If we wanted to, we could file 10 suits a day" against bulletin boards, Saunders says. Instead, the association contacts the systems operators to let them know they will soon be monitored, thus giving them a chance to remove pirated software.

Even fears of contracting viruses has not discouraged many bulletin board users from picking up duplicated software.

Software rental firms are another target of industry watchdogs. Firms renting software, even if it is a legal operation, exacerbate software piracy, says Ronald Palenski, general counsel of Adapso.

To curtail this threat, legislation is under consideration by the House and the Senate that will give software vendors the right to restrict rental of their software. If it passes, the legislation, which mirrors rental restrictions for record companies, places the onus of risk on the vendor. This will mean, in effect, that software vendors themselves will decide if their software can be copied.

One got away

Despite trade association efforts, the piracy picture is not completely clear. A precedent-setting case — and one that has aroused the ire of the software industry — is one in which state agencies were allowed to pirate software without paying damages.

BV Engineering v. *the University of California at Los Angeles* was first brought to district court in 1987, decided in 1988 and denied a hearing by the U.S. Supreme Court in 1989.

The case determined that the owners of BV Engineering, whose software had been pirated by the engineering department at UCLA, had no standing to sue for damages. The difference between the state and others, Saunders says, is that the state can receive an injunction to stop the piracy but is not currently liable for damages, court costs and attorney's fees.

Concerned that other state agencies will be encouraged to pirate software, last year industry advocates introduced remedial legislation to the Copyright Act of

How to spot pirate software

To determine whether advertised software is pirated, it is best to use the adage that "anything that sounds too good to be true probably is," says Peter Beruk at Software Publishers Association (SPA).

The first telltale sign of pirated software is the price. Most major software programs advertised for less than $200 are probably illegal, Beruk says.

The second warning signal is the location of the mail-order house. "Stay alert for Hong Kong, the Middle East and the Far East," warns Pilar Cloud, executive assistant at Washington, D.C.-based Business Software Association (BSA). Once a catalog is received, consumers can call vendors' regional representatives to find out whether the source is a authorized distributor.

Because unauthorized duplication of software costs, vendor associations encourage wary consumers to contact them to obtain more information or report suspected piracy.

BSA can be reached at 202-737-7060; SPA's piracy hot line is 1-800-388-PIRA.

1976 to the U.S. Congress. The goal, Palenski explains, "is to close the state agency loophole."

Dave Eskra, chairman of both Adapso and Pansophic Systems, Inc., in Lisle, Ill., testified before the Senate Subcommittee of Patents, Copyrights and Trademarks that since the decision, it has become harder to close deals with state agencies on mainframe software sales because the agencies have no incentive to license the product.

Palenski says he expects the legislation to be passed in early 1990, thus thwarting piracy by state organizations, although, he adds, "In this town, I wouldn't be surprised by anything."

Foreign piracy

Beyond U.S. shores, software pirates on both halves of the hemisphere have been thick as thieves. The retail and mail-order piracy market in the Far East has recently been compounded with corporate piracy cases in Hong Kong. In Europe, corporate piracy has become widespread in certain countries.

Spurred by a maturing microcomputer base in foreign countries and amendments to the Trade Act, the software industry has linked hands with the book publishing, recording and motion picture associations under the auspices of the International Intellectual Property Alliance (IIPA), formed in 1985.

Before 1984, the Trade Act contained provisions that provided duty-free entry of goods normally tariffed into developing nations. It also did not expressly protect copyrighted intellectual property of any sort.

With a mature market in installed bases of microcomputers, videocassette recorders and inexpensive copying technology, the heads of intellectual property associations found rampant copyright infringements during a trip to the Far East.

Hence, the allied industries filed a report with the U.S. Trade Representative office, which passed an amended Trade Act explicitly to protect intellectual property in foreign countries. If the countries do not comply, the U.S. Trade Representative has the right to invoke trade sanctions.

The IIPA and its individual members, including BSA, have been working in tandem with the European Commission (EC) on a software protection directive.

If adopted, the directive will require the 12 EC member states to amend their laws to protect all software used in their country. The EC's Parliament and the Counsel of Ministries are expected to consider the directive in the first quarter of 1990.

"Essentially, it will protect the concept that when you open a shrink-wrapped package of software, it implies that a contract has been entered," says Douglas E. Phillips, president of BSA in Washington, D.C.

Recently, BSA has filed several suits in France and Italy and has issued a warning in Spain. Last April, in a raid of Montedison, an $11 billion Italian corporate conglomerate, investigators found that 90% of Lotus Development Corp. and Ashton-Tate Corp. software was illegally copied. Nine months later, the case is still in progress.

Two French companies, against which BSA filed suit last October, include Telediffusion de France, a provider of transmission services to the broadcast media, and Banque Paribas, a financial institution.

In Spain, BSA has tried different tactics by announcing to the press that the group is planning to conduct a raid against a major Spanish corporation. "Every company in Spain could potentially be raided," Phillips says, "so this is a warning to them to stop using pirated software."

Phillips hopes this strategy works, because, as he points out, BSA members — which include Aldus Corp., Ashton-Tate, Autodesk, Inc., Lotus, Microsoft Corp. and Wordperfect Corp. — "are in the business of developing software, not litigation."

In the Far East, cases have been filed in Hong Kong, Taiwan and Korea, among other countries. And BSA is working with the People's Republic of China to enforce stronger copyright laws.

Pirated ship comes in

The biggest success story, perhaps, is Singapore. Phillips reports that a year ago, "pirated software was carried by plenty of retail stores." After raids and lawsuits against six major retail outlets and an uncovering of an international mail-order operation, he says, the pirated software trade has really dried up.

By working closely with international governments and trade associations to avoid "being seen as Americans coming in and telling people how to behave," Phillips says he is confident that international piracy will continue to be stymied.

To think that piracy will be completely eliminated is unrealistic. A more attainable goal is to have it perceived as risky conduct.

"We want to get to the point," Phillips says, "where stealing software from the store and copying it are seen as the same thing." •

Casualties of war

Recent events in the war on piracy:

- The Business Software Association (BSA) initiates criminal proceedings against Mapfre Vida, a Spanish insurance firm, for alleged illegal software copying. *Jan. 8, 1990.*
- BSA announces legal action against Fox, S.A., of Barcelona, Spain, for copyright infringement stemming from alleged use of unauthorized copies of software made by Ashton-Tate Corp., Lotus Development Corp., Microsoft Corp. and Wordperfect Corp. *Dec. 14, 1989.*
- National Institute of Justice publishes a new resource manual on computer crime. It is aimed at helping auditors, security experts and criminal justice agencies combat various computer crimes, including piracy. *Dec. 1, 1989.*
- Singapore police and BSA raid five targets of suspected software piracy. *Nov. 24, 1989.*
- BSA announces it will take legal action against a Hong Kong piracy syndicate that allegedly copied $50 million worth of programs and manuals. *Nov. 17, 1989.*

MINI POLL

Handling a tough issue

How do you handle the sticky issue of illegal software duplication? We asked that question of some IS chiefs:

DONALD WHITTINGTON
MIS Manager
Michigan Sugar Co.
Saginaw, Mich.

We have a written policy that forbids unauthorized copying. However, having the policy is one thing, enforcing it another. In many cases, it's hard to track the originator, and what can you do if someone comes in after hours and makes copies?

If we did find employees copying software, we'd warn them against continuing to do so. I've found, however, that since we've put our policy in writing, illegal copying has not been a problem.

CHRIS PORCH
Vice-President
Computer Operations
Pacific Century Advisors
San Diego

We do not have a formal policy for dealing with duplication, but being part of a large bank — Security Pacific Bank — we've got auditors in here once or twice a year to take inventory of the hard drives. The much larger issue we're worried about is viruses. But I do warn people that pirating software is not condoned. It's not the individual that will get fired, it's me. Large organizations like ours could get stiff penalties if copying occurred. I think the courts are geared toward hanging corporations.

We buy a lot of software, and I think site licenses for corporations our size would go a long way in helping us combat piracy.

ALLEN HEAD
Manager
Information Services
Universal Flavor Corp.
Indianapolis

We have a formal company policy stating, "Thou shalt not duplicate software," and that policy is well-enforced. If we catch someone a first time, we'll warn him. For a second offense, we'd have to take more drastic action.

People are not restrained from buying software for their work. If they need Lotus' 1-2-3, we'll buy it for them. That encouragement helps deter software piracy.

Memo: Computer Viruses and Personal Computers

Sandra Bogenholm, CSSO
DOE Center for Computer Security

This was originally written for the N-4 group at Los Alamos National Laboratory. CSSOs may wish to edit it to fit local conditions and use it to educate personal computer users at their sites.

As a personal computer user you are responsible for certain data protection functions on your PC or workstation system that are handled automatically by the system manager of a mainframe system. These functions include providing physical security for the system and media, providing adequate backup for all software and data, ensuring that only information appropriate to the authorized levels of classification and category is stored or processed on your system, and ensuring that infected software is not run on your computer.

Recently, many viruses (or related code) have been infecting computer systems around the laboratory. A virus is a "self-propagating Trojan horse, composed of a mission component, a trigger component, and a self-propagating component." * A virus can cause a number of benign or serious problems anything from a message on the screen, to data alteration, data loss, etc.

The most likely entry point for a virus is at the microcomputer level. From there it can spread to other micro or mainframe computers to which the microcomputer is networked or with which you share media. As a PC user, you are our most important line of defense against a costly and embarrassing virus infection. By keeping your system and media free from viral infection, you protect not only yourself but also users with whom you share files. Most of us share files with the office word processors and other staff, so let's practice "Safe Computing." If all of us take the responsibility to protect our own systems and media, we will all be protected.

Attached is a list of guidelines to help you minimize the likelihood of a virus infection, diagnose the presence of a virus, and respond in the event of an infection. A Telephone Call Checklist (similar to a bomb threat telephone checklist) is included to help you conduct an interview with anyone phoning to threaten or inform you of a virus attack. Please copy the checklist and keep it in your phone book along with the bomb threat instructions.

Computer Virus Guidelines

Protection from viral infection includes knowing your software sources and limiting sources to commercial ones whenever possible. It also includes limiting access to your computer and its media. Recovery from infection is facilitated by having backups of the operating system, application programs, and data files and by your keeping several previous backups so that you are sure you can go back to a point before the infection to reconstruct the system and data. Finally, knowing your system and running virus detection programs helps you monitor your system to ensure contamination-free files and system.

Preventative and Damage Control Measures

A. Backup

Make frequent data file backups and store the diskettes or other media in a safe location (ideally in a different office and building from your computer). Files that would be difficult or time-consuming to recreate should be backed up most often. Practice recovering your files from the backups. There are commercial software programs that can quickly back up your hard disk. Save the backup diskette sets of critical or hard-to-replace files for at least a year unless they become obsolete before then.

Always make a backup or working copy of application software; never run directly from the distribution disk. If you have problems with your disk, computer, or a virus, you can reinstall the software after the problem is corrected.

Never boot your system with the original operating system diskettes. Make backup copies before you install the system software, and use them for intallation. Write-protect and store the original diskettes in a safe location. Subsequently, boot from the hard disk or from your backup copies. Also, never add data or programs to the original system diskettes.

The best possible protection against virus infection damage or other disk problems is a correct and thorough backup procedure. At the very least, anyone using a PC (Mac or IBM, etc.) should make backup copies on removable disks of all data files and application programs.

B. Software

1. Unauthorized or Noncommercial Software

Reprinted with permission, from the August 1989 issue of Center for Computer Security News. Department of Energy, Center for Computer Security.

Do not bring ANY unauthorized code (software) into the workplace especially software downloaded from public bulletin boards. Be suspicious of any software or software media supplied by friends. It is recommended that software be purchased through normal procurement channels or that it be reviewed by knowledgeable programming/security personnel.

Do NOT use shareware when a commercial product is available. Try to obtain source code whenever possible.

If you believe it is necessary to use noncommercial software, limit your sources to the most established, reliable ones. A colleague down the hall may or may not be a reliable source because he is unlikely to have checked the software thoroughly (unless he wrote the application himself), even if he has used it for some time. (Remember, some viruses have a time bomb or usage count detonator embedded in them).

If you must use noncommercial software,

- Try to get the source code (not just the executable code). - Have an experienced programmer/security person do a security review of the code and investigate anything suspicious. - Ask for software design documentation and reviews if appropriate. - Do anything else you can to ensure the safety of the code.

Beware of files you create on your home computer and bring to work. Has someone used the home computer and imported a virus? Family members may have added infected shareware or games obtained from friends or bulletin boards. Or the neighborhood whiz kid may have planted an original or copied virus on your machine.

2. Software Development

Assign sensitive software development tasks to trusted persons or subject all software to independent review before it is installed.

Use a two-person rule for software and hardware design, implementation, testing, and review. Better yet, encourage the use of good software engineering practices and hold design reviews, code walk-throughs, etc. Keep development and production isolated from each other.

Associate each copy or module of software with an individual who is responsible for it.

3. Specific Systems

When you initially install your operating system software (DOS or MS-DOS users), examine the COMMAND. COM file. Write down its size, creation date and creation time. Periodically reexamine COMMAND.COM to see if any changes in size, date, or time have occurred. Such changes may mean that a virus has corrupted the file. If you note unexplainable changes, rebuild DOS with the "SYS" command.

All file servers or networked Sun, Apollo, and other computers running Unix should have anonymous FTP disabled and the sendmail utility installed without debug. You should recheck these after major operating system upgrades.

Check any multiuser system to ensure that all anonymous, debug, dealer service, and other general user identifiers, passwords, and accounts are disabled. These should also be checked after each operating system upgrade.

There are programs available for both Macintoshes and IBM PCs that can be run periodically to look for known virus behaviors. You should run such a program at least once a month, but preferably more often. Ask your CSSO or computer security organization about such detection programs.

4. Write Protection and File Locking

If your operating system supports locking files to prevent changes (easy to do for Macintosh, Unix, and VMS), set that protection on all files that you seldom change. Although a virus can get round this, many have not been written to anticipate locked files so some protection is provided by taking this precaution.

When you obtain new software, write-protect the distribution disk or tape before making a backup or working copy or installing the software on your hard-disk. Just inserting a disk in an infected system can be enough to corrupt the diskette. In fact, it is always a good idea to use write-protected disks unless you know you will need to write to a disk.

When you use commercial software, try to avoid packages that use copy protection. This allows you to follow the preceding suggestion. Applications and data can be kept on write-protected removable media (cartridge drives or floppies) and inserted into a workstation only when needed.

5. Miscellaneous

Test every unknown program before system-wide release (preferably on an isolated system).

Use password security if available.

Report any unauthorized use of your system to your CSSO.

If you believe your system has high vulnerability to a virus, contact your CSSO or computer security organization. They may be able to aid you in the use of virus detection software. Make it a practice to power down your microcomputer overnight, and do not leave diskettes in the disk drives overnight. You may wish to install a locking device on the power switch to prevent unauthorized access to your computer.

Have standard recovery procedures in place. Now is a good time to develop a contingency plan.

Diagnosing the Presence of a VIrus

The best way to detect the presence of a computer virus is to be as familiar as possible with the way your computer runs in daily operation. In addition, look for the following indications of system contamination:

Program or data files mysteriously disappear.

Unusual messages appear on the screen. Some viruses even announce that your system has been infected.

An unusual number of program or system crashes or print errors occur.

Sudden, unexplainable reductions in system memory or disk space occur.

Your computer seems to run more slowly than normal.

Program loads take longer than normal.

An unusual number of disk accesses occur.

Disk drive access lights come on for no apparent reason.

An executable file, particularly COMMAND. COM, changes in size.

Unexplainable hidden files appear. IBM PC-DOS V4.0 has three hidden files, earlier versions have two. But be aware that some application software does create legitimate hidden files.

On a Macintosh some icons (in particular, those representing the Scrapbook and Notebook) change in appearance.

Responding if You think You have a Virus

Record all you can about the circumstances and details.

If a strange message appears on the screen, record the EXACT content of the message.

Do nothing irrevocable. Do not reformat the disk. Do not turn the system power off.

Ask yourself: Is my computer attached in any way to another computer? If so, should this connection be broken until the problem is solved?

Try to isolate the hardware and software that you suspect is infected.

Contact your CSSO for help. If the CSSO is not available, contact your supervisor or computer security organization, and notify the CSSO as soon as possible.

Prevent the transmission of any suspected software across any network.

Try to establish the source by thinking about where your software came from, who has been using your machine, etc.

Don't try suspicious software on another system use a completely isolated and cleansed system and only if you know what you are doing.

The advice in this article is based in part on information kindly offered by the Kansas City Computer Virus Team at Allied Signal, Inc., R. K. Wallace, X-DO, Los Alamos National Laboratory and Jared Dreicer, DOE Center for Computer Security, LANL.

Computer Virus Telephone Checklist

Time_____
Date _____

1. Ask the caller's name.

_____
_____

2. Try to determine if the caller is offsite or within the organization.

_____
_____

3. Try to get another person on the line with you.

4. Ask what computer system or what type of computer it will affect.

_____
_____

5. If the caller says it is a virus, ask the following questions:

A. Where is the source of attack for the virus-network, phone port, imported software, etc.

_____
_____

B. What will the virus do to show its presence; what is its ultimate effect?

_____
_____

C. Why is the virus here?

_____

D. What will disarm the virus?

_____
_____

6. Note background noises on the line-music, traffic, motors, or unusual sounds.

_____
_____

7. Think about the caller:

Male _____
Female _____

Any accent or speech impediment?

_____
_____

Anything familiar or unusual?

_____
_____

8. When the call is completed:

A. Call your CSSO or computer security organization.

B. If appropriate, call the DOE Center for Computer Security (FTS) 843-0444 or (505) 667-0444.

C. Tell your supervisor

TURING AWARD LECTURE

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

Originally published in *Communications of the ACM*, Vol 7, Number 8, August 1984.

KEN THOMPSON

INTRODUCTION

I thank the ACM for this award. I can't help but feel that I am receiving this honor for timing and serendipity as much as technical merit. UNIX[1] swept into popularity with an industry-wide change from central mainframes to autonomous minis. I suspect that Daniel Bobrow [1] would be here instead of me if he could not afford a PDP-10 and had had to "settle" for a PDP-11. Moreover, the current state of UNIX is the result of the labors of a large number of people.

There is an old adage, "Dance with the one that brought you," which means that I should talk about UNIX. I have not worked on mainstream UNIX in many years, yet I continue to get undeserved credit for the work of others. Therefore, I am not going to talk about UNIX, but I want to thank everyone who has contributed.

That brings me to Dennis Ritchie. Our collaboration has been a thing of beauty. In the ten years that we have worked together, I can recall only one case of miscoordination of work. On that occasion, I discovered that we both had written the same 20-line assembly language program. I compared the sources and was astounded to find that they matched character-for-character. The result of our work together has been far greater than the work that we each contributed.

I am a programmer. On my 1040 form, that is what I put down as my occupation. As a programmer, I write programs. I would like to present to you the cutest program I ever wrote. I will do this in three stages and try to bring it together at the end.

STAGE I

In college, before video games, we would amuse ourselves by posing programming exercises. One of the favorites was to write the shortest self-reproducing program. Since this is an exercise divorced from reality, the usual vehicle was FORTRAN. Actually, FORTRAN was the language of choice for the same reason that three-legged races are popular.

More precisely stated, the problem is to write a source program that, when compiled and executed, will produce as output an exact copy of its source. If you have never done this, I urge you to try it on your own. The discovery of how to do it is a revelation that far surpasses any benefit obtained by being told how to do it. The part about "shortest" was just an incentive to demonstrate skill and determine a winner.

Figure 1 shows a self-reproducing program in the C[2] programming language. (The purist will note that the program is not precisely a self-reproducing program, but will produce a self-reproducing program.) This entry is much too large to win a prize, but it demonstrates the technique and has two important properties that I need to complete my story: 1) This program can be easily written by another program. 2) This program can contain an arbitrary amount of excess baggage that will be reproduced along with the main algorithm. In the example, even the comment is reproduced.

[1] UNIX is a trademark of AT&T Bell Laboratories.

```
char s[ ] = {
        '\t',
        '0',
        '\n',
        '}',
        ';',
        '\n',
        '\n',
        '/',
        '*',
        '\n',
        (213 lines deleted)
        0
};

/*
 * The string s is a
 * representation of the body
 * of this program from '0'
 * to the end.
 */

main( )
{
        int i;

        printf("char\ts[ ] = {\n");
        for(i=0; s[i]; i++)
                printf("\t%d, \n", s[i]);
        printf("%s", s);
}
Here are some simple transliterations to allow
    a non-C programmer to read this code.
=       assignment
==      equal to .EQ.
!=      not equal to .NE.
++      increment
'x'     single character constant
"xxx"   multiple character string
%d      format to convert to decimal
%s      format to convert to string
\t      tab character
\n      newline character
```

FIGURE 1.

STAGE II

The C compiler is written in C. What I am about to describe is one of many "chicken and egg" problems that arise when compilers are written in their own language. In this case, I will use a specific example from the C compiler.

C allows a string construct to specify an initialized character array. The individual characters in the string can be escaped to represent unprintable characters. For example,

"Hello world\n"

represents a string with the character "\n," representing the new line character.

Figure 2.1 is an idealization of the code in the C compiler that interprets the character escape sequence. This is an amazing piece of code. It "knows" in a completely portable way what character code is compiled for a new line in any character set. The act of knowing then allows it to recompile itself, thus perpetuating the knowledge.

Suppose we wish to alter the C compiler to include the sequence "\v" to represent the vertical tab character. The extension to Figure 2.1 is obvious and is presented in Figure 2.2. We then recompile the C compiler, but we get a diagnostic. Obviously, since the binary version of the compiler does not know about "\v," the source is not legal C. We must "train" the compiler. After it "knows" what "\v" means, then our new change will become legal C. We look up on an ASCII chart that a vertical tab is decimal 11. We alter our source to look like Figure 2.3. Now the old compiler accepts the new source. We install the resulting binary as the new official C compiler and now we can write the portable version the way we had it in Figure 2.2.

This is a deep concept. It is as close to a "learning" program as I have seen. You simply tell it once, then you can use this self-referencing definition.

STAGE III

Again, in the C compiler. Figure 3.1 represents the high level control of the C compiler where the routine "com-

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
...
```

FIGURE 2.2.

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
...
```

FIGURE 2.1.

```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return(11);
...
```

FIGURE 2.3.

pile" is called to compile the next line of source. Figure 3.2 shows a simple modification to the compiler that will deliberately miscompile source whenever a particular pattern is matched. If this were not deliberate, it would be called a compiler "bug." Since it is deliberate, it should be called a "Trojan horse."

The actual bug I planted in the compiler would match code in the UNIX "login" command. The replacement code would miscompile the login command so that it would accept either the intended encrypted password or a particular known password. Thus if this code were installed in binary and the binary were used to compile the login command, I could log into that system as any user.

Such blatant code would not go undetected for long. Even the most casual perusal of the source of the C compiler would raise suspicions.

The final step is represented in Figure 3.3. This simply adds a second Trojan horse to the one that already exists. The second pattern is aimed at the C compiler. The replacement code is a Stage I self-reproducing program that inserts both Trojan horses into the compiler. This requires a learning phase as in the Stage II example. First we compile the modified source with the normal C compiler to produce a bugged binary. We install this binary as the official C. We can now remove the bugs from the source of the compiler and the new binary will reinsert the bugs whenever it is compiled. Of course, the login command will remain bugged with no trace in source anywhere.

```
compile(s)
char *s;
{
        ...
}
```

FIGURE 3.1.

```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

FIGURE 3.2.

```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

FIGURE 3.3.

MORAL

The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code. In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. A well-installed microcode bug will be almost impossible to detect.

After trying to convince you that I cannot be trusted, I wish to moralize. I would like to criticize the press in its handling of the "hackers," the 414 gang, the Dalton gang, etc. The acts performed by these kids are vandalism at best and probably trespass and theft at worst. It is only the inadequacy of the criminal code that saves the hackers from very serious prosecution. The companies that are vulnerable to this activity, (and most large companies are very vulnerable) are pressing hard to update the criminal code. Unauthorized access to computer systems is already a serious crime in a few states and is currently being addressed in many more state legislatures as well as Congress.

There is an explosive situation brewing. On the one hand, the press, television, and movies make heros of vandals by calling them whiz kids. On the other hand, the acts performed by these kids will soon be punishable by years in prison.

I have watched kids testifying before Congress. It is clear that they are completely unaware of the seriousness of their acts. There is obviously a cultural gap. The act of breaking into a computer system has to have the same social stigma as breaking into a neighbor's house. It should not matter that the neighbor's door is unlocked. The press must learn that misguided use of a computer is no more amazing than drunk driving of an automobile.

Acknowledgment. I first read of the possibility of such a Trojan horse in an Air Force critique [4] of the security of an early implementation of Multics. I cannot find a more specific reference to this document. I would appreciate it if anyone who can supply this reference would let me know.

REFERENCES

1. Bobrow, D.G., Burchfiel, J.D., Murphy, D.L., and Tomlinson, R.S. TENEX, a paged time-sharing system for the PDP-10. *Commun. ACM 15*, 3 (Mar. 1972), 135–143.
2. Kernighan, B.W., and Ritchie, D.M. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, N.J., 1978.
3. Ritchie, D.M., and Thompson, K. The UNIX time-sharing system. *Commun. ACM 17*, (July 1974), 365–375.
4. Unknown Air Force Document.

Author's Present Address: Ken Thompson, AT&T Bell Laboratories, Room 2C-519, 600 Mountain Ave., Murray Hill, NJ 07974

The Science of Computing

The Internet Worm

Peter J. Denning

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 to generate 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 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 still 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 had 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 as authorities pondered legal questions. Morris himself was silent throughout.

The worm's author went to great lengths to confound the discovery and analysis of it, a delaying tactic that permitted the massive infestation. By early December 1988, Eugene Spafford of Purdue (*3*), Donn Seeley of

Peter Denning is Director of the Research Institute for Advanced Computer Science at the NASA Ames Research Center.

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 were 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 in 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, and then broke the connection. If the 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 of over 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 took 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 a system, and it frequently changed its process identifier.

Originally published in *American Scientist*, Vol 77, March-April 1989.
Reprinted by permission.

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. They were impressed by 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 in 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 or 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 the 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 had 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.

We should acknowledge 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

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:

—Every account should be protected by a password.

—Passwords should be stored in an enciphered form, and the file containing the enciphered passwords should not be publicly accessible (it is in UNIX).

—Passwords should be deliberately chosen so that simple attacks cannot work—for example, they could include a punctuation mark and a numeral.

—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.

—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.

—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.

—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.

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, we must bear in mind that the Internet is an open network, whereas 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. The risk is compounded by "mutually trusting hosts," a design in which a group of workstations is treated 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 concluded 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 behavior 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, telephones, 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 of networks of computers. We cannot separate technology from responsible use.

References

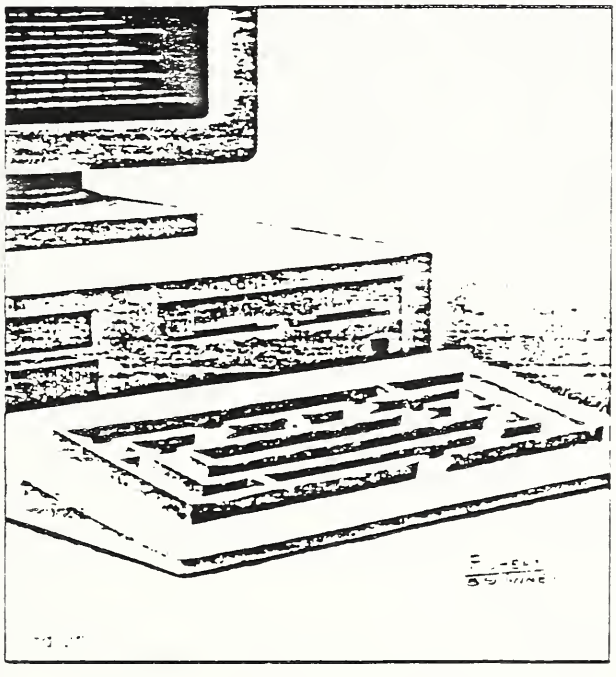
1. E. Marshall. 1988. Worm invades computer networks. *Science* 242: 855–56.
2. E. Marshall. 1988. The worm's aftermath. *Science* 242:1121–22.
3. E. Spafford. 1988. *The Internet Worm Program: An Analysis*. Tech. rep. no. CSD-TR-823, Comp. Sci. Dept., Purdue Univ. Also published in *ACM Comp. Commun. Rev.*, Jan. 1989.
4. D. Seeley. 1988. *A Tour of the Worm*. Tech. rep., Comp. Sci. Dept., Univ. of Utah. Also published in *Proc. Winter Usenix Conf.*, Feb. 1989. Usenix Assoc.
5. M. Eichin and J. Rochlis. 1988. *With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988*. Tech. rep., MIT Project Athena.
6. B. Kocher. 1989. A hygiene lesson. *Commun. ACM* 32:3, 6.
7. J. F. Shoch and J. A. Hupp. 1982. The worm programs—Early experience with a distributed computation. *Commun. ACM* 25: 172–80.
8. P. J. Denning. 1988. Computer viruses. *Am. Sci.* 76:236–38.
9. F. T. Grampp and R. H. Morris. 1984. UNIX operating system security. *AT&T Bell Labs Tech. J.* 63:1649–72.

Secret Codes

Any good data security system must rely on encryption

Asael Dror

Cryptography is the ancient art of making the comprehensible incomprehensible to all but a chosen few—of keeping secrets secret. Julius Caesar is credited with protecting the secrecy of messages by replacing every letter in the original text, called the *plaintext*, with a letter three characters later in the alphabet. The result is called a *ciphertext*, in which *A* is represented by *D*, *B* by *E*, and so on.



The war between cryptographers, who devise *cryptosystems*, and code breakers, who try to decipher encrypted messages, has drastically escalated since the invention of the computer. On one hand, computers help to break complicated cryptosystems within seconds. On the other hand, they make it feasible to use extremely complex encryption algorithms that were not practical before. Furthermore, the advent of distributed computer systems, the wide availability of microcomputers, advances in mass storage, and the widespread use of computer communications have all contributed to moving cryptography from military and diplomatic fields to those of more general interest and importance.

Two major cryptosystems are in use today: conventional systems and public-key systems. Two major encryption algorithms relate to these cryptosystems: DES and RSA, respectively.

Conventional Cryptosystems

One important method of encryption is *substitution*: replacing every occurrence of a letter (or word, or byte) with a different letter (or word, or byte). The XOR operator is a convenient way to perform substitution with computers. When you XOR 2 bits together, the result is 1 if one and only one of the input bits is 1. The result is 0 if both input bits are 0, or if both input bits are 1.

The XOR function is convenient because it's fast and you can decrypt the encrypted information simply by XORing the ciphertext with the same data that you used to encrypt the plaintext. For example, you can encrypt the word *TEST* by XORing every byte with the ASCII representation of the letter *A* (0100 0001). In figure 1a, the letter *A* is the *key* used to encrypt the plaintext. To decrypt the message, you XOR it again with the same key, as in figure 1b.

The strength of a good cryptosystem doesn't depend on keeping its algorithm secret; the secrecy of the ciphertext relies solely on the secrecy of the key.

A *statistical cryptoanalysis* attack can easily break a simple cryptosystem. Natural language has specific known patterns, such as the frequency with which each letter is used; common letter combinations, such as *th*, *er*, *ing*, and *ion*; and

continued

word-usage frequency. Those plaintext patterns will also appear—although their expression will differ—in the ciphertext: once you recognize them, you can use them to break the cipher. Alternately, you can break the cryptosystem with a *brute-force* attack. Since there are only 256 possible keys (binary 0000 0000 to 1111 1111), a computer can quickly try them all.

One way to overcome these problems is to use longer keys. For example, you could use a four-"letter" key such as A5GE (a good "random" key). In this case, you encrypt the first byte with *A*, the second byte with *5*, the third with *G*, and the fourth with *E*. After exhausting the key, you reuse it; so you encrypt the fifth byte using *A* again, and so forth. The key length is 4, making it harder, but not impossible, to use letter-pattern methods to break the code.

Unfortunately, if code breakers know (or can guess) part of the plaintext (e.g., if they know that every message begins with "Dear Sir"), then they can use *analytical cryptoanalysis* to derive the key. In figure 1, XORing the plaintext with the ciphertext reveals the key.

Ideally, you should have a key that never repeats. Such a key, composed of random bits and never reused, is called a *one-time tape* (or *one-time pad*). You can prove mathematically that a cryptosystem based on a one-time tape is unbreakable. Unfortunately, such a cryptosystem requires a key as long as the message you want to encrypt; so then you have to figure out how to transmit the key safely. Still, one-time-tape systems are usable when a safe transportation means is available now but won't be in the future, when you need to transmit the secret message.

It may seem that you could create a one-time-tape cryptosystem by extending a short key with a computer's random-number-generating function, using the short key as the seed. Although many commercially available data-encryption packages use such a scheme, it should be considered more of a toy than a cryptosystem. A computer's random-number generator actually generates pseudo-random numbers. A mathematical relationship exists between each generated number and the one that follows it. Consequently, such proprietary cryptosystems, often described as unbreakable, can usually be cracked within minutes (see reference 1).

DES

Since a layman cannot tell the difference between a secure cryptosystem and a complete mockery, the National Bureau of Standards (NBS) established the Data Encryption Standard (DES). It was originally developed by IBM and adopted as a standard by the NBS in 1977. In 1980, it was adopted by ANSI.

Prior to becoming a standard, the security of DES was validated by the National Security Agency (NSA), which found the algorithm free of any statistical or mathematical weaknesses (see reference 2). Since its adoption as a standard, DES has been used by most banks for money transfers and by most U.S. government agencies (except the military).

DES works on one 8-byte (64-bit) block at a time. The encryption process is controlled by a user-supplied 56-bit key; that's 2^{56} (72,057,594,037,927,936) possible keys. Every bit in the output is a complex function of every bit in the input block and every bit in the key. Decryption under DES is the reverse of encryption and is performed by working the algorithm backward. The encryption process (see figure 2) consists of an initial permutation of the input block followed by 16 rounds of encipherment, and finally an inverse of the initial permutation.

After the initial permutation, the block being encrypted is divided into two parts, called L_0 and R_0. In each of the 16 rounds of encipherment (see figure 3), the new L part is the previous round's R part. The new R is the previous round's L part XORed with the result of the *cipher-function f*. Thus, the output of round *i* is

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i)$$

The cipherfunction *f* (see figure 4) derives its output based on the old R part (R_{i-1}) and the current round's key (K_i). You use the inputs to perform substitution via eight lookup tables called S boxes and then permute the combined output of the S boxes to give the function's output.

Each round uses a different 48-bit key, K_i. You derive the current round's keys by performing a set of permutations and left shifts on the user-supplied 56-bit key. DES defines the exact left shifts and permutations used to derive each round's key, as well as the definition of the S boxes and all the other required permutations (see reference 3).

The strength of DES has been ascertained by the NSA's thorough analysis and years of widespread use without any known break in the system. DES's biggest weakness is its limited key length. Its critics claim that you might be able to break DES with a brute-force attack (i.e., by trying every possible key).

However, trying all possible keys within a reasonable time frame would require a special machine that would use as many as 1 million processors working concurrently. Each processor would decrypt the ciphertext using a different set of keys and check (e.g., by using a dictionary) to see if it had guessed the correct key. Even though it would cost millions of dollars to construct such a machine (if at all feasible), the fact that DES is in such common use creates an incentive to develop it. However, this theoretical shortcoming is no reason not to use DES. If you are worried about it, you can easily

(a)				
Plaintext: (*TEST*)	0101 0100	0100 0101	0101 0011	0101 0100
Key: (the letter *A*)	0100 0001	0100 0001	0100 0001	0100 0001
Ciphertext: (plaintext XOR key)	0001 0101	0000 0100	0001 0010	0001 0101
(b)				
Ciphertext:	0001 0101	0000 0100	0001 0010	0001 0101
Key: (the letter *A*)	0100 0001	0100 0001	0100 0001	0100 0001
Plaintext: (ciphertext XOR key)	0101 0100	0100 0101	0101 0011	0101 0100

Figure 1: *A simple example of* (a) *encryption and* (b) *decryption using the XOR operator*

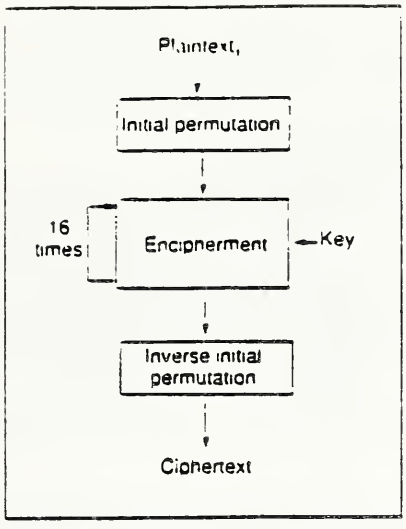


Figure 2: *The DES encryption process. Every bit in the output is a complex function of every bit in the input block and every bit in the key.*

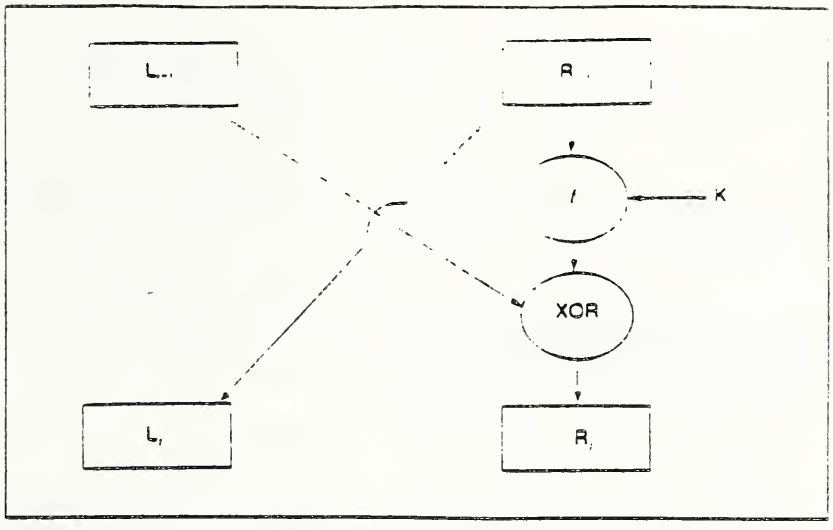


Figure 3: *The details that are involved in each of the 16 rounds of encipherment shown in figure 2. Note that the new L part is the R part from the previous round, and the new R is the L part from the previous round XORed with the result of cipherfunction f.*

overcome it by using an additional pre-DES encryption stage.

Public-Key Cryptosystems

When using a conventional cryptosystem such as DES, both the sender and the receiver must know the key used to encrypt (and decrypt) the data. Therefore, you need a safe means of transmitting the key from one to the other. If you change the keys frequently, transmitting them becomes a major problem. Furthermore, with a conventional cryptosystem, it's impossible to communicate with someone new until you have safely exchanged keys; this can take a long time. Public-key cryptosystems are designed to overcome these shortcomings.

Public-key cryptosystems are based on the use of a *trap-door one-way function*. You can easily compute such a function in one way only—used to encrypt the data. To compute the function in the other direction—used to decrypt the data—you must have certain secret information; hence, the name *trap-door*.

In a public-key cryptosystem, each person has two keys: one for encrypting, E_x, and one for decrypting, D_x. Decrypting with D_x a plaintext P that was encrypted using E_x restores the original plaintext—that is, $D_x(E_x(P)) = P$. Both E_x and D_x should be easy to compute, but knowing E_x does not reveal D_x.

If you use a public-key cryptosystem, you can publish your encrypting key E (the public key) in a public directory, while you keep D (the private key) secret. If someone wants to send you a message, all that person has to do is look up your public key (E_x) and use it to encrypt the message as $E_x(P)$. Only you

continued

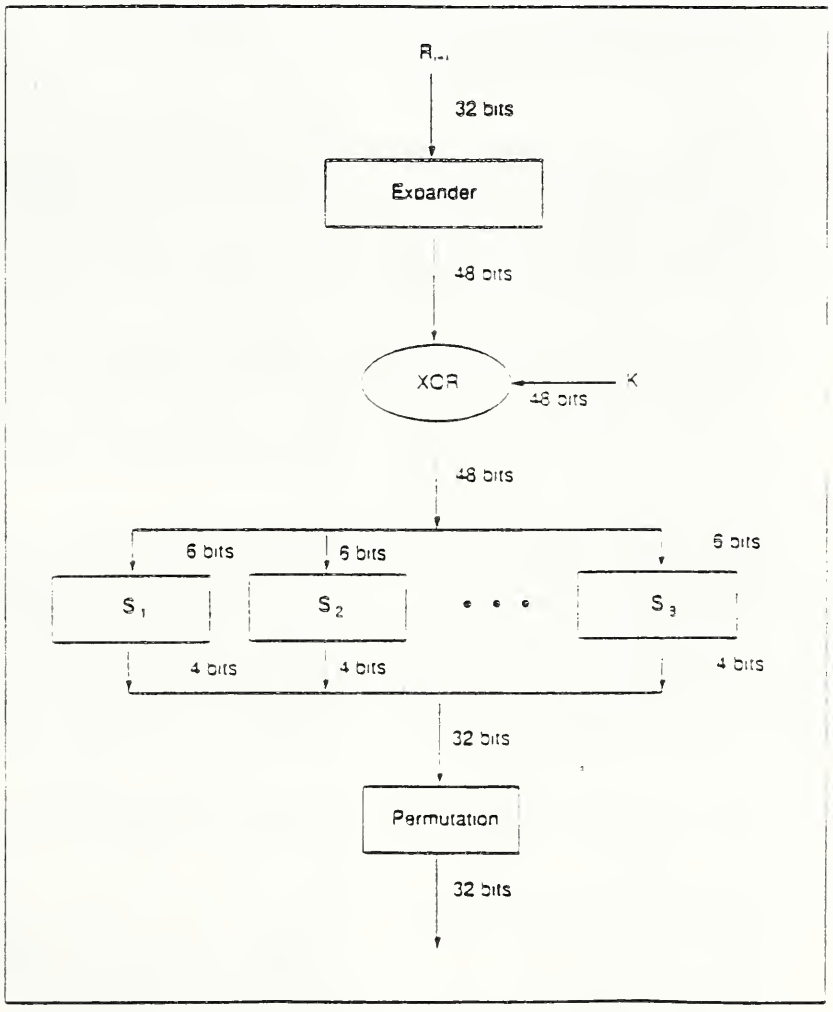


Figure 4: *Cipherfunction f from figure 3. Its output comes from the old R part (R_{n-1}) and the current key (K_n). The inputs use eight S boxes to perform substitution and then permute their combined output to give the function's output.*

know the private key D_A, so only you can decrypt the message back to its original plaintext, $D_A(E_A(P)) = P$.

RSA

The most important public-key cryptosystem today is RSA (see reference 4), named after its inventors, Rivest, Shamir, and Adleman. To use RSA, you need to choose, at random, two large prime numbers, to be called p and q. Compute n as the product of the two primes: $n = p*q$. Then, randomly choose a large number d, so that d is *relatively prime* to $(p-1)*(q-1)$; in other words, the greatest common divisor of d and $(p-1)*(q-1)$ is 1. Finally, compute e so that $(e*d)$ modulo $((p-1)*(q-1)) = 1$. The notation "x modulo y" signifies the remainder of dividing x by y using integer division. For example, 20 modulo 5 = 0, since 20/5 = 4 with 0 remainder; 13 modulo 3 = 1 since 13/3 = 4 with 1 remainder.

The public key is the pair of numbers (e,n), and the private key is (d,n). Although n and e are public, it is difficult to arrive at d, since there is no efficient algorithm for factoring large numbers. Consequently, to be secure, both p and q must be very large (at least 100-digit numbers), so that n is extremely large (at least 200 digits) and cannot be factored within a reasonable time.

To encrypt with RSA, first you break the plaintext into blocks that can be represented as an integer between 0 and $n-1$. Then, you encrypt each block by raising it to the power e, modulo n. To decrypt the block, raise it to the power d, modulo n; that is, $C = P^e$ modulo n, and $P = C^d$ modulo n.

Let's look at an example of how to use RSA. For the sake of simplicity, you should use very small primes for p and q. To create a secure system, however, you should use very large primes (to find large prime numbers, see reference 5).

- Assume you choose $p = 3$ and $q = 11$.
- Then, $n = p*q = 3*11 = 33$ and $(p-1)*(q-1) = 2*10 = 20$.
- You can use $d = 7$, since 7 is relatively prime to 20.
- Next, you need to find an e, so that $e*7$ modulo 20 = 1.
- You can use $e = 3$ because $3*7 = 21$, and 21 modulo 20 = 1.
- Thus, your public key is (3,33) and your private key is (7,33).

If you represent your message by using a 1 for *A*, 2 for *B*, 3 for *C*, and so on, the plaintext DEAD would be written as 4 5 1 4. The following table shows how to encrypt this using the public key (3,33).

P	P^e	P^e modulo n
4	64	31
5	125	26
1	1	1
4	64	31

Thus, the ciphertext is 31 26 1 31 (using large primes would let you create large blocks that would conceal the patterns detectable in this simplified example).

To decrypt this, you would use the following to restore the original plaintext.

C	C^d	C^d modulo n
31	27512614111	4
26	8031810176	5
1	1	1
31	27512614111	4

The RSA algorithm has been known since 1978, and in no known case has it been broken. Its strength is based on the complexity of factoring very large numbers. However, while no algorithm has yet been found to efficiently factor large numbers, such an algorithm may exist. If such an algorithm is found, RSA would be rendered useless. Furthermore, no one has proven that factoring n is essential to deriving the private key.

On a more practical note, RSA's operations on very large numbers make the system extremely slow. In addition, the RSA algorithm is patented, and you can't use it freely.

Digital Signatures

In addition to ensuring privacy, encryption can be used to verify authenticity. Say you send your broker a message telling him to sell all your stocks. How can the broker verify that you sent it? If you dispute ever sending the message, how can the broker prove that you did? If you used paper mail, your signature would be used to verify and prove authenticity, but how about electronic messages?

Simply encrypting the message using a key known only to you and the broker doesn't solve the problem. The broker would be satisfied that you had sent the message, but couldn't prove it since he knows the key and thus could have forged the message. Public-key cryptosystems can provide an elegant and simple solution by creating *digital signatures*.

A trap-door one-way function has the property of $D(E(P)) = P$. If the function used by the public-key cryptosystem also has the property of $E(D(P)) = P$, it is said to be a *trap-door one-way permutation*. The RSA public-key cryptosystem fulfills this requirement. Using such a public-key cryptosystem, you can encrypt the message using the private key D_A. Anyone who receives the message $D_A(P)$ can decrypt it using your public key E_A, since $E_A(D_A(P)) = P$. Since D_A is known only to you, the recipient knows, and can prove, that you are the author.

If you want to send a private message that can be authenticated to someone else, then you encrypt $D_A(P)$ with that person's public key, giving $E_B(D_A(P))$. Using the private key, D_B, that person would derive $D_B(E_B(D_A(P))) = D_A(P)$, which would be saved as proof of authenticity, and then decrypt $D_A(P)$ by using $E_A(D_A(P)) = P$. Thus, both privacy and authenticity have been achieved.

Secure Computer Systems

Any good computer data security system must rely on encryption. Whereas both DES and RSA provide a good basis for a computer security system, using proprietary algorithms may be worse than using no encryption at all, because they lead to a false sense of security.

But encryption alone is not sufficient. Proper key selection, key management, physical security, people security, and procedures to ensure that plaintext does not "leak" out of the system via loopholes (see reference 6) are all essential for a secure computer data system. ■

REFERENCES

1. Kochanski, M. "A Survey of Data Insecurity Packages." *Cryptologia*, January 1987, pp. 1–15.
2. Meyer, C., and S. Matyas. *Cryptography: A New Dimension in Computer Data Security*. New York: John Wiley & Sons, 1982.
3. National Bureau of Standards. *Data Encryption Standard*. Federal Information Processing Standard Publication 46, 1977.
4. Rivest, R., A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." *Communications of the ACM*, February 1978, pp. 120–126.
5. Knuth, D. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1969.
6. Dror, A. "Data Protection and Encryption." *The Waite Group's MS-DOS Papers*. Howard W. Sams, 1988, pp. 217–239

Asael Dror is the founder and president of Wisdom Software, located in San Francisco, California. He is the author of File Encrypt, a DES encryption program for MS-DOS and OS/2. He can be reached on BIX as "asael."

READING LIST
OF
SELECTED COMPUTER SECURITY ARTICLES

This reading list is the product of the Information Exchange Working Group of the Cmputer and Telecommunications Security Council, a government/industry technical group that was sponsored by NIST from 1987 to 1990. The entries provide titles, sources, and reviews of important publications.

Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation. **Staff study by the Subcommittee on Investigations and Oversight, Committee on Science, Space, and Technology. U.S. House of Representatives. August 3, 1989. 35 pages.**

This report is useful because of the many useful references cited and the short, emphatic quotes that are effective in presentations on the need for a formal role to be played by information security specialists in the software development and acquisition process.

The detailed report comes from a study of current literature, cases of software failure, and interviews of over 50 experts including Dr. Peter Neumann from SRI. It documents the disastrous state of software quality and reliability including security. Software development methods are described including the failure of the waterfall method of user requirements specification followed by implementation. The need for an iterative process, e.g., Boehm's Spiral method, is emphasized along with recognizing problems of costing this approach. Major software disasters are cited.

A good presentation on the need for security in the products and development efforts is made but no solutions are offered. Ethics and certification of programmers are also discussed. Further study of the problem is recommended.

The Cuckoo's Egg, by Clifford Stoll. Doubleday 1989. 326 pages.

This is a personnel history of the life of a computer system manager during an 18-month siege by a malicious hacker. It is not a technical treatment. (See Dr. Stoll's ACM Communications article May, 1988 for a technical approach.) The title derives from the Cuckoo bird's practice of putting its eggs in other birds nests to hatch. The book's most interesting story describes the metamorphosis of a 1960s Berkeley activist to a responsible adult with respect for law and order. The book's strong message is the need to reject and eject malicious hackers from the community of computer network users. Chapter 55 is a good case study of the German hackers who worked for the KGB. Otherwise, the book makes light and enjoyable reading.

Final Report on Computer-Related Crime, Council of Europe, Publications Division, Strasbourg, France or Manhattan Publishing Co., P.O. Box 659, Croton, New York 10520. 1989. 82 pages.

Robert E. Smith (Privacy Journal) reports this document as one of the most methodical and useful studies of legal issues in computer crime. It contains guidelines for national legislation on computer crime as well as evidentiary, procedural, and other problems of transnational offenses.

Computer Crime: Criminal Justice Resource Manual, Second Edition, by Donn B. Parker. NCJ 118214 (National Institute of Justice, 800-851-3420, P.O. Box 6000, Rockville, MD 20850). 1989. 220 pages.

This manual, a rewritten edition of the original manual published in 1978, is written for investigators and prosecutors of computer crime in the U.S. However, it provides insights on this subject important for information security specialists. In particular, advice is provided on the requirements for security detection controls to produce information acceptable as criminal evidence. Analyses of federal and several state computer crime statutes are included.

Organizing for Computer Crime Investigation and Prosecution, by Catherine H. Conly. NCJ 118216. National Institute of Justice. 1989. 124 pages.

Dedicated Computer Crime Units, by J. Thomas McEwen. NCJ 118215. National Institute of Justice. 1989. 129 pages.

Information Technology Installation Security, Federal Systems Integration and Management Center, Office of Technical Assistance, 5203 Leesburg Pike, Suite 400, Falls Church, VA 22051. December 1988. 98 pages.

This publication addresses all government managers having responsibility for information technology. The publication is intended to assist them in developing, implementing, and maintaining security policies, procedures and techniques.

Computer Viruses: Dealing with Electronic Vandalism and Programmed Threats, by Eugene Spafford, Kathleen Heaphy and David Ferbrache. ADAPSO, 1300 North Seventeenth Street, Suite 300, Arlington, VA 22209. 1989. 109 pages.

This book presents a high-level discussion of computer viruses, explaining how they work, who writes them, and what they do. It is not a technical reference on viruses. The book dispels common myths about viruses and provides simple, effective suggestions on how to protect computer systems against threats.

Managing Information Resources: New Directions In State Government, Dr. Sharon L. Caudel, Dr. Donald A. Marchand with Dr. Stuart I. Bretschneider, Ms. Particia T. Fletcher, and Mr. Kurt M. Thurmaier. School of Information Studies, Syracuse University, 4-206 Center for Science and Technology, Syracuse, NY 13244-4100. August, 1989. 307 pages.

This report is a joint effort between the National Association for State Information Systems, Inc., information processing industry companies, and Syracuse University's School of Information Studies which directed the research. The principal objectives of the study were to inventory and analyze the management policies and practices applied to information and information technology in state governments and to share those approaches.

Security in Open Systems: A Security Framework (ECMA TR/46), European Computer Manufacturers Association, 114 Rue du Rhone, 1204 Geneva, Switzerland. July 1988. 71 pages.

This technical report provides a framework for the development of standards that support a wide variety of security requirements in a multi-user, multi-vendor systems environment. The report gives an overview of security requirements from both the operational and the functional point of view. It also gives implementation considerations and design requirements relevant to the design of secure systems on the basis of this framework.

***Security in Open Systems Data Elements and Service Definitions (Alice in Wonderland),* European Computer Manufacturers Association, 114 Rue du Rhone, 1204 Geneva, Switzerland. July 1989. 74 pages.**

This ECMA Standard defines data elements and services for the support of a wide variety of security requirements in a multi-user, multi-vendor distributed system environment. The data elements and services developed in this standard are based on concepts defined in ECMA/TR46: "Security in Open Systems - A Security Framework."

***DOE Risk Assessment Instructions, Resource Tables, and Completed Sample -- A Structured Approach,* Department of Energy's Office of ADP Management and the Computer and Technical Security Branch (Raymond Barrow 301-353-3307). Two volumes and one diskette. September 1989.**

The guideline presents a simplified, structured approach to the risk assessment process. When completed, the risk assessment results produce an Executive Summary which provides a mechanism for briefing, reviewing, and discussing the identified risks with upper management and assists with the identification of resources needed to implement appropriate countermeasures.

Some Technical Security Hazards Associated with Copier and Printer Technologies, Defense Copier Security Working Group of the Defense Information Security Committee. November 6, 1989. 9 pages.

This report highlights and discusses several technical security hazards identified during the work of the Defense Copier Security Working Group of the Defense Information Security Committee. The hazards exist in printers used as automated information system hard copy output devices as well as in copiers. The report is not all inclusive; it focuses primarily on copier security procedures and some limited, basic security features that support them. The report does address the major security hazards inherent in this equipment.

Guide for Selecting Automated Risk Analysis Tools (Special Publication 500-174), Irene Gilbert. National Institute of Standards and Technology. October 1989. 26 pages.

This document recommends a process for selecting automated risk analysis tools. It is intended primarily for managers and those responsible for managing risks in computer and telecommunications systems. The document describes important considerations for developing selection criteria for acquiring risk analysis software. The information presented is derived from reviews of risk analysis software tools in the Risk Management Laboratory which is cooperatively sponsored by the National Institute of Standards and Technology and the National Computer Security Center and from experiences of organizations in the federal government and private sectors.

Computer Security Training Guidelines (Special Publication 500-172), Mary Anne Todd and Constance Guitian. National Institute of Standards and Technology. November 1989. 32 pages.

This document provides a framework for identifying computer security training requirements for a diversity of audiences who should receive some form of computer security training. It focuses on learning objectives based upon what computer security knowledge is required by an individual in their job. The guide divides employees involved in the management, operation, and use of computer systems into five audience categories.

Executive Guide To The Protection Of Information Resources (Special Publication 500-169), **Cheryl Helsing, Marianne Swanson, Mary Anne Todd. National Institute of Standards and Technology. October 1989. 14 pages.**

This guide is designed to help the policy maker address a host of questions regarding the protection and safety of agency computer systems and data processed. It introduces information systems security concerns, outlines the management issues that must be addressed by agency policies and programs, and describes essential components of an effective implementation process.

Management Guide To The Protection Of Information Resources (Special Publication 500-170), **Cheryl Helsing, Marianne Swanson, Mary Anne Todd. National Institute of Standards and Technology. October 1989. 14 pages.**

This guide describes the issues that must be addressed by agency managers in meeting their responsibilities to protect information resources within their organizations. It outlines the critical elements of an information resource protection process that applies to a stand-alone personal computer or to a large data processing facility.

Computer User's Guide To The Protection Of Information Resources Special Publication 500-171), Cheryl Helsing, Marianne Swanson, Mary Anne Todd. National Institute of Standards and Technology. October 1989. 12 pages.

This guide makes the computer user aware of some of the undesirable things that can happen to data. It provides practical solutions for reducing the risks to such threats as unauthorized modification, disclosure, and destruction, either deliberate or accidental.

Data Encryption Standard Fact Sheet, Computer Security Division, National Institute of Standards and Technology, (301) 975-2934. January 1990. 9 pages.

This document addresses those frequently asked questions regarding various aspects of the Data Encryption Standard (DES). It provides interested individuals with sources of additional information. The document does not issue new policy; rather it summarizes and clarifies existing policies.

Computers under Attack: Intruders, Worms, and Viruses, Peter Denning, Editor, ACM Press, 1990. Order Number 706900. $17.50. 150 pages in paperback. (ACM Press, 11 West 42nd Street, New York, NY 10036. Tel (212) 869-7440, fax (212) 944-1318).

From the advertisement: "A collection of some of the most informative, provocative, and frightening reports on the vulnerability of computer systems to harmful, if not catastrophic, attacks. Denning is editor-in chief of Communications of the ACM and is Director of the Research Institute for Advanced Computer Science, NASA-Ames Research Center.

Spectacular Computer Crimes, Buck Bloombecker. Dow Jones Irwin, Homewood, IL 60430. 242 pages.

This is an interesting book that expresses the unusual legal ideas of the author about famous computer crimes. The use of real names of victims and perpetrators in several of the very old, settled cases described does a disservice to those people and organizations who should have the right to outlive those painful experiences.

Commitment To Security, **National Center for Computer Crime Data, (NCCCD, 904 Daniel Court, Santa Cruz, CA 95062) 1989.**

This report was sponsored by ISSA and several companies by the not-for-profit NCCCD and is the product of J.J. Buck Bloombecker and Dr. Stanley Stahl. Copies were sent to all ISSA members free of charge. It contains the best and worst of statistics about information security.

Disaster Recovery Journal, **Richard L. Arnold, publisher, 2712 Meramar Drive, St. Louis, MO 63129.**

Contingency Journal, **Bob Thomas, publisher, 10935 Estate Lane, Suite 375, Dallas, TX 75238.**

Crisis Magazine, **Robert J. Bogle, publisher, 190 South Warner Road, Suite 100, Wayne, PA 19087.**

These are three new trade journals in the disaster recovery and contingency planning field that has become a specialty field in its own right.

Systems Auditability and Control Reports, **Institute of Internal Auditors, 249 Maitland Ave., Altamonte Springs, FL 32701. Future publication.**

In February 1990, the Institute of Internal Auditors' Research Foundation (IIA) began a year-long comprehensive revision of the definitive Systems Auditability and Control (SAC) Reports that it published in 1977. The original report resulted from a study undertaken for the Foundation by SRI International. The revision is under the direction of Price-Waterhouse and is expected to be completed in mid-1991. The project is expected to cost more than $1.25 million. $500,000 has already been obtained from IBM. IIA plans call for expanding the three volumes of the earlier SAC edition into 10 modules and for supplementing these with seminars and video-based training materials.

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4545

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
April 1991

4. TITLE AND SUBTITLE

National Institute of Standards and Technology Internal Report (NISTIR)
Computer Security: Selected Articles

5. AUTHOR(S)

Marianne Swanson, Elizabeth Lennon

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
NIST Internal Report (NISTIR)

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

NIST
Computer Security Division

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The NISTIR contains nine computer security articles and a reading list of computer security publications. This information will benefit computer security managers as well as managers and users of information technology.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

encryption; internet; local area network; risk analysis; security; viruses

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
43

15. PRICE
A03

ELECTRONIC FORM