WARNING
The following material may be
INFECTIOUS!

By the time you read these words, computer virus–related diseases will have infected more than a million computers from PCs to giant mainframes, causing everything from minor annoyance to the crashing and burning of entire systems.

Your bank account, millions of daily transactions on Wall Street, credit-card accounts totalling billions of dollars, airline reservations affecting hundreds of thousands of passengers daily, even the programmed instructions of American nuclear missiles—all are potentially at the mercy of criminals, secret agents, terrorists, or even malicious pranksters who know how to access the systems and inject their own deadly “computer virus.”

What do we do? Who do we call to protect our computer resources?
In *Virus!,* viral expert Allan Lundell presents you with:

- A detailed analysis of the structure and workings of the infamous Internet Worm virus
- Virus tales from the cyberpast
- Interviews with the minds behind the viruses
- Biological counterparts to the computer viruses
- New paradigms of thinking and uses for these viral programs
- Cures, antidotes, advice, and warnings about antiviral "experts"

No one is invulnerable. No system is guaranteed secure. At no time in history has the power of one destructive individual been so amplified.
Dedicated to our home planet Earth and to her continuing splendor and unfolding beauty
CONTENTS

Preface ix
1 The Internet Virus Story 1
2 Origin of the Species 19
3 Modern History 40
4 Victims of the Virus 72
5 Viral Politics 76
6 Inside the Viral Mind 87
7 Biological Counterparts 109
8 Viral Cures 121
9 Viral Futures 130
Postscript 140
Appendix A: Analysis of the Internet Worm 145
Appendix B: A Software Bestiary 161
Appendix C: Methods of Distributing Computer Viruses 167
Appendix D: How Antiviral Programs Work 170
Appendix E: Recommended Reading for Virus Watchers 176
Glossary 177
Acknowledgments 188
ABOUT COMPUTER VIRUSES:

"The dangers of viruses are just unbelievable. . . . The threat is more serious than most people think."
—Donald Latham, Executive Vice President, Computer Sciences Corporation and former Assistant Secretary of Defense for the Reagan administration

"Most mainframe computers can be successfully subverted within an hour. Huge international networks with thousands of computers can be opened up to an illicit intruder within days."
—Dr. Fred Cohen, Professor, University of Cincinnati, and creator of computer viruses since 1983

"It's like a fantasy of being a terrorist without the blood."
—Eric Corley, Editor, 2600, a national hacker newsletter
"A virus is deadly because it can jump—actually slide right through the barriers everyone uses to control access to valuable information."

—Kenneth Weiss, Technical Director, Security Dynamics Technologies

“It’s apparently going to be the game this year: to see who can come up with the best virus. We are all very vulnerable.”

—Dennis Steinaur, Senior Security Specialist, National Bureau of Standards

“Viruses are really serious stuff for software publishers. As a publisher of software, we have to be sure that nothing gets onto our master disks, and as a business that uses Macintosh, we have to protect our information and our machines from infection.”

—Charlie Jackson, President, Silicon Beach Software

“Some of the possibilities are so horrifying I hesitate to set them down at all.”

—A. K. Dewdney, Professor, University of Western Ontario and creator of Core War

“We may be able to reprogram biological viruses to help rather than harm us.”

—Dr. Expansion, Hacker to the MMMaaxx

“Ha! Ha! Ha! You’re finished, all of you!! It won’t be long before I’m in charge!! I can hardly wait. . . .”

—L. Luthor, Master Criminal of Planet Earth
“Ultimately, there is no security.”
—Dr. Fred Cohen, Professor, University of Cincinnati, and creator of computer viruses

“And ‘ultimately’ may be sooner than you think.”
—Allan Lundell, author of Virus!

By the time you read these words, computer virus–related diseases will have infected over a million personal computers worldwide, causing everything from minor annoyances to complete loss of data. The dreaded network viruses have disabled thousands of computers, causing massive data devastation and the literal crashing and burning of some unfortunate systems. Robert Morris, Jr.’s ground-breaking Internet Worm virus almost seems like a fond memory compared to the beasts roaming today’s cyberspaces. But not to fear, unless you so desire. What do we do? Who do we call to protect our computer resources?
In this book, I present you with:

- Detailed analysis of the infamous Internet Worm virus
- Virus tales from the cyberpast
- Interviews with the minds behind the viruses
- Legal actions and control issues concerning viruses
- Biological counterparts to the computer viruses
- New paradigms of thinking and uses for these viral programs
- Cures, antidotes, and solutions to the viruses
- Future viral scenarios

Greetings, and welcome to the microworld of computer viruses.

I have often wondered what the author of the first book on human disease thought when trying to express to his readers the ideas of human disease and microscopic invaders. Just think: he had to take people from their primal fear of disease (created by watching friends and neighbors keel over and die inexplicably) to recognizing that the illness was actually caused by tiny little "demon creature" invaders.

Following in that author's illuminated text-steps, I also apply a principle that has worked throughout time for our species: shed some light on the item creating fear, and, presto, our fear recedes. It is amazing how well the reduction of ignorance eliminates fear.

Researching the story of the computer viruses—who thought them up, when they were created, where they came from, what they've done so far, how they work, and where they are likely to head—has been quite illuminating. It was a great thing to do for the last eight months. Now, as I finish this book, the nation is besieged by the first really big network virus, the Internet Worm virus, rendering thousands of computers useless within a few hours.
What happens when entire infrastructures of industrialized cultures (like ours) become dependent on computers and computer networks, and computer diseases begin to wreak havoc on the electronic grid? How do we deal with status-seeking programmers who want to make the six o'clock news by knocking out the IRS?

What do we do? Who do we call? Well, there are "bug busters" out there now, and they are listed in the Appendixes. And there are viral protection mechanisms that can be installed on your computer. Everything from viral condoms to computer sterilizations are discussed in Chapter 8. The Appendixes also list available protection hardware and software.

One of my favorite chapters in Virus! is Chapter 6, "Inside the Viral Mind." Who are these guys who are creating computer viruses? Why do they do it? What's their payoff? Does it really matter who they are? How important is it, anyway, to think about the causes of the disease? Is what we find out about the viral mind really useful in controlling their future actions?

Has thinking about the causes of crime really made a difference in the incidence of criminal attack? Should we focus more on simply neutralizing the disease? What do we really want to accomplish here anyway?

In the computer world, the last question is pretty simple. Most computer people want their machines to be disease-free, the way it was for most of them last year. Who needs this new abomination, a problem almost guaranteed to take us out of our high-tech Garden of Eden? Besides, we have enough problems as it is—issues of software and hardware compatibility, piracy, memory shortages, Japanese domination. Who needs disease?

Well, I saw a bumper sticker the other day that simply says, "Shit Happens." It seems as if the most workable attitude toward viruses is simply to take what happens in stride and see what we can do with it. After all, many of us are problem solvers by nature, and just maybe computer diseases, specifically computer viruses, are actually useful in some way.
A major distinction between biological disease and computer disease is that the latter is a human-engineered process. That fact means that we can fully understand how to create the disease mechanism.

In biology we are constantly studying processes more complex than our understanding of how they work. Researchers in the biological arena spend a lot of time trying to figure out the details of the organic processes they are observing.

In computer science, it is just the opposite; we are creating processes based on our existing understanding. In this arena, we don't simply study and analyze phenomena. If we want to fully understand a virus, for example, we create one. And in that creation process, we gain some mastery of what we have created.

With computer science, we are more a cause, creating reality, rather than being an effect of it. Being the cause is like being God; we are playing an active role in creating the movie we find ourselves in. The computer is our genuine creation, our evolving child, even if at present its IQ is no more than that of a grasshopper. Knowledge is always increasing, and our child will continue to evolve, interconnecting with us in more and more ways.

Even in their relatively primitive state of evolution, computers have become the backbone, an essential component of the infrastructure of our culture—keeping things humming on a daily basis.

The evolution of the computer is also yoked to our evolution as a species, and to our understanding of health, illness, and well-being. And one of the things we are discovering is that disease, as a life process, is profound, especially when we consider that we are now creating and releasing diseases designed to afflict our very own brainchild. With computer viruses, our thoughts, especially destructive ones, can easily be turned into action.

I have been studying the evolution of computers and high technology for the last eight years. As an editor of
and contributor to many computer publications, including *InfoWorld*, *Byte*, *Popular Computing*, *Corporate Times*, *Computer Currents*, *Digital Deli*, *HyperAge*, and *MacWorld*, I've seen microcomputers evolve from the status of toys to an essential tool of our play and workplace today. I have a strong love for these machines, as they have provided me many hours of entertainment, knowledge, and work, enriching my life beyond all expectations. They have facilitated my communicating with the world around me, and they have allowed me to live in beautiful places without the need to fight rush-hour traffic.

Computer viruses have caught my attention for many reasons. The foremost one is obvious—they threaten my livelihood and lifestyle. And because the technology they infect is so widespread, viruses threaten anyone else who uses microcomputers as tools of the trade. My background in the biological sciences contributes to my interest and has proved invaluable in writing this book.

It is my hope to convey to you, the reader, the scope of the computer virus story without bogging you down in technical difficulties. There is light at the end of the tunnel, but it is a light that requires an interesting journey to reach. Enjoy, and feel free to communicate with me any ideas that you feel may be relevant.
VIRUS!
On November 2, 1988, during the final week of the Bush-Dukakis presidential contest, the most virulent virus ever to hit the computer world made its debut on planet Earth. In less time than it takes the sun to travel from horizon to horizon, the Internet virus wormed its way at blistering speeds into the memories of over six thousand computers scattered nationwide.

An explorer ship designed to probe the vast Internet labyrinth full of juicy academic and military data, the artificial creature never reached its full potential, falling ill to a bug in its programming near the beginning of its journey. We begin our story with its launch into the Internet cyberspace:

The time was 9 p.m. at the MIT artificial intelligence laboratory in Cambridge, Massachusetts. Acting on a remote signal originating from Ithaca, New York, the gleaming new version 1.0 probe was launched from its hard disk "holding pen" into a telephone line.

Its destination: the Internet.
Its purpose: widespread exploration and infection of the labyrinth without detection.

Its design: streamlined, equipped with several penetration devices especially designed for bypassing Internet security barriers.

As the virus entered Internet cyberspace for the first time, it easily passed the entry tests of the Internet boundary guards, showing them the electronic equivalent of an "Internet technician" pass. The pass basically said, "This user has official permission to work on our Sendmail electronic mail system and our UNIX Finger Daemon program. He has high-priority access."

If the pass hadn't worked, the probe would have flashed to the electronic guards thousands of possible password IDs with a good probability that one of them would have gained entry.

Logging onto Internet was difficult. System response was very sluggish, as if the system were extremely busy with computer traffic so late at night. The probe was everywhere, wantonly making copies of itself and sending these copies in all directions within the labyrinth. The copies would then replicate themselves thousands upon thousands of times, rapidly filling up all the empty space in the Internet system.

At about ten o'clock that night, Pascal Chesnais, a computer researcher working late night at MIT's Media Laboratory in Cambridge, Massachusetts, noticed something very odd about the behavior of his computer programs. Everything on the screen had slowed to a crawl. Two or three of his friends also working observed the same bizarre behavior.

At first, they figured a legitimate program had gone out of control because of some internal error. "We thought it was just a runaway program," he recalls. "So we killed all the processes, started over, and the problem seemed to go away." Unconcerned, they went out for ice cream.

Meanwhile, out on the West Coast, at 10:15 EST, the Experimental Computing Facility at the University of California at Berkeley was penetrated by the probe.
Berkeley's new experimental security software was detecting bizarre behavior on the network communication lines. "Our security system sent us alerts that it was receiving unusual commands," recalls Peter Yee, a scientist at the center.

This early warning allowed the Berkeley crew to contain the virus faster than any other node on the Internet grid. The Berkeley crew was on it. Not only did they stop the probe from further replication by shutting off the communications link to most of their computers, but they established a "trap" to capture the probe, so that it could be studied and analyzed.

Meanwhile, researchers at Bellcore, the Livingston, New Jersey, joint research laboratory for the regional Bell holding companies, discovered the virus at 10:30 P.M. They, too, were able to contain the virus by quickly shutting down computers.

At 10:34 P.M., the invader struck Princeton University, though no one really noticed the strike until midnight. Victor Dukhovni, a twenty-five-year-old systems programmer, was getting ready to go to bed; just before going to sleep, he turned to his home computer and called up the mathematics department computer, asking for a backup of files. It was a task he performs every day, part of his job servicing the department's machine.

Victor, too, noticed strange things going on. The system was slow, and it was running unrecognizable programs. Alarmed, Victor dressed and embarked on a three-minute journey across the quiet campus to the math department. A relative newcomer at his job, he did not yet possess the home phone numbers of other department members who could help. Working alone, Victor identified the probe in the mail system, reproducing itself at a rapid rate. He wondered what to do.

By this time, the little creature had wormed its way into NASA's Ames Research Center in California's Silicon Valley, as well as into the University of Pittsburgh and into the Los Alamos National Laboratory in New Mexico.

At midnight EST, Ames researchers cut off all commu-
nications with outside researchers, more than fifty-two thousand computer users. Though Los Alamos officials knew something odd was going on, they did not suspect a major viral infection for several hours.

It was also about midnight when Pascal and friends at MIT returned from their ice cream break. Much to their dismay, they found their computers were acting, once again, very sluggish. “It was as if twenty or thirty people were working at the same time on the system,” Pascal says. “Then we knew something was amiss.” With the whole crew crowding around two terminals, the programmers began tracing the virus.

Meanwhile Robert T. Morris, Jr., a twenty-three-year-old Cornell University computer graduate student, telephoned a friend at Harvard’s Aiken Laboratory and asked him to send out an alert over the network with instructions as to how to disable the virus. The friend did write a brief message in technical language explaining how to stop the virus from spreading, ending the message with the comment, “Hope this helps, but more, I hope this incident is a hoax.” The friend’s message, unfortunately, was accidentally sent to an obscure electronic bulletin board, to be seen by few, if any, researchers.

At 12:31 Thursday morning, the virus hit Johns Hopkins University in Baltimore, and at 1:15 A.M., the University of Michigan in Ann Arbor.

By 2:30 in the morning, Pascal announced, “We have definitely determined that something is coming through the electronic mail. We have shut ourselves off the network so the virus in our system won’t spread.”

Exhausted, Pascal and the MIT team went home at 3:30 A.M., knowing they had a serious problem with no immediate solution.

By then, the Berkeley team had specifically identified the virus—but they had no antidote either. In his 2:28 A.M. alert, Mr. Yee warned, “We are currently under attack from an Internet Virus. . . . The program appears as files that start with the letter ‘x.’ Removing them is not enough, as they will come back in the next wave of
attacks. For now, turning off the [mail] seems to be the only help."

What made the virus particularly terrifying at the time was that nobody knew for sure that it was benign. "There's no reason why the virus couldn't have wiped out people's files or put subtle time bombs in their systems," says Robert Logan, a computer systems manager at California Institute of Technology.

Meanwhile, the Berkeley crew maintained their virus watch, feverishly seeking a cure to the network invader. "There were about a dozen people working in a small room on eight computers and terminals," Scott Mclvev, a twenty-three-year-old Berkeley senior from Cupertino, California, told the New York Times. "The space was crowded. The phones were ringing. People called from the Navy, from the Air Force, and from Florida."

"The callers were discovering this program that was replicating itself and doing strange things," says David Wasley, a UC-Berkeley computer manager. "A number of copies of the virus were running on the computer simultaneously. . . . Each copy was trying to identify all the other computers that that computer knows about, trying to discover the passwords of all the users."

Some computers connected through Internet weren't infected, although not for lack of trying by the virus. At the University of Maryland, for example, a computer equipped with a security system logged about two thousand failed attempts by the persistent pest. (Others at the school without such protection were infected.) The virus attacked Argonne National Laboratory outside Chicago, starting at 11:54 P.M. EST Wednesday and continuing throughout the night. But only one of the lab's many computers was infected. Luckily, researchers a few months before had modified the widely used UNIX code for gaining access to the system.

Interestingly enough, AT&T's Bell Laboratories in Murray Hill, New Jersey, where young Robert Morris, Jr., had worked for a time, escaped infection. About a year prior to the attack, Bell Labs had patched its software to elimi-
nate the loophole in the electronic-mail software. When Bell had tried to warn other groups of UNIX users of the potential for a breach in security, Bell found that few shared "our rather paranoid view of communications software," says M. Douglas McIlroy, a member of the technical staff.

Classified defense computers were not affected by the attack, even though Arpanet (within Internet) is used for unclassified, defense-related work. Fortunately, U.S. defense computers employ greater security than unclassified systems, making the defense computers more difficult to penetrate.

As researchers figured out how the virus worked, they began calling friends and colleagues around the country. Hans-Werner Braun, an engineer at the University of Michigan in Ann Arbor, was awakened at 4:00 A.M. by virus trackers in California. Heading for the home computer connected with his computer at work, Mr. Braun applied the virus fix within an hour.

Berkeley posted a bulletin reading:

Newsgroups: news.announce.important
Subject: UNIX TCP/IP Virus (ADMINISTRATORS READ THIS IMMEDIATELY)

Description:

There's a virus running around: the salient facts. A bug in sendmail has been used to introduce a virus into a lot of Internet UNIX systems. The Virus has not been observed to damage the host system, however, it's incredibly virulent, attempting to introduce itself to every system it can find. The virus appears to use rsh, broken passwords, and sendmail to introduce itself into the target systems. The virus affects only VAXes and SUNs, as far as we know.

There are three changes that we believe will immunize your system. They are attached. Thanks to the Experimental Computing Facility, Center for Disease Control, for their assistance.
But communications among researchers were limited by the fact that they often deal with one another by electronic mail, not by telephone. With many computers shutting down to isolate the virus, nationwide electronic mail was seriously disrupted. And with this disruption, messages like the preceding one did not get delivered to every party concerned.

In reaching some of the stricken computer sites, the Berkeley crew found that many had no emergency phone contacts or contingency plans to cope with such an outbreak. "The sites without an emergency plan didn't do well," says Russell Brand, an artificial intelligence doctoral candidate at Berkeley and a researcher at the nearby Lawrence Livermore Laboratory. He happened to be writing a paper on computer security on his PC when he was informed of the outbreak.

The standard voice telephone became the main communications tool for these cyberworld doctors. Soon they began identifying the structure of the virus. The E-mail that did get through showed that the probe was being deciphered rather quickly, though the creature was tricky to catch:

>From ulysses!arpa!purdue.edu!spaf

The virus seems to consist of two parts. I managed to grab the source code for one part, but not the main component (the virus cleans up after itself so as not to leave evidence).

Its behavior patterns were noted by the MIT crew:

This virus tries to find all the machines it can talk to, copies itself into those machines, and then the virus starts doing the same thing again. From disassembling the code, it looks like the programmer is really anally retentive about checking return codes.

By the early afternoon of Thursday, November 3, 1988, the viral code had been cracked, its wild breeding held in
check. Slowly, one by one, the computers linked to Inter­
net were brought back on line, with the data flow soon returning to normal levels. Everyone involved with the incident, however, would never return to normal, knowing firsthand what the future may hold for us all.

Though most experts agree that penetration of a top­
secret classified computer system would be extremely difficult, skeptics have pointed out that the Internet virus managed to infect the unclassified portion of a second network, called Milnet, which also carries secret information.

ARPAnet is named after the Advanced Research Pro­
jects Agency, a Pentagon research agency whose com­
puter network is used in most research centers in the nation. Internet is a larger system that includes much of ARPAnet and Milnet.

Within days, investigators identified Robert Morris, Jr., as the probable source of this virus. What is this creature that Morris allegedly designed?

The Internet virus is actually more like a worm than a virus. Strictly speaking, a virus is a program that adds itself to other programs, including operating systems. A virus cannot run independently, requiring a host pro­
gram to activate it.

The worm is a program that is self-contained; it runs by itself and can propagate a fully working version of itself from machine to machine. The basic object of the Internet worm was to get into other machines so that it could further propagate itself.

The worm had three ways by which it penetrated through machine security: the Sendmail attack, the Fin­
gerd attack, and password cracking.

In the Sendmail attack, the worm entered the com­
puter systems through a back door in a Sendmail elec­
tronic mail utility that had been left there deliberately by the program’s designer.

The worm/virus also made use of a little-known com­
mand embedded within a utility program called Fingerd. Fingerd operates like a telephone book or information
operator. It is used for identifying the names, phone numbers, and addresses of system users and when they were last on the network. The Fingerd program is designed to run as a background or daemon process, a service on your electronic desk that can be collecting data for you while you are doing other things on the network. The worm/virus took advantage of Fingerd by overrunning the daemon's input buffer, by bombarding it with too much information too fast. This caused the daemon to malfunction allowing the worm to slip past the "electronic information operator" and into the local system for which the operator was responsible.

The worm also asked each computer encountered whether or not the computer had already been infected. If not, it would worm its way into that computer. Apparently it was this part of the worm that caused it to breed out of control. To get around false infected signals, the worm was designed to copy itself to the host machine after ten queries, no matter what answer it received. This caused infected computers in the Internet system to be reinjected thousands of times, causing a massive proliferation of worms in each computer. Computer scientists who analyzed the code say the designer should have had the worm make at least ten thousand queries (instead of ten) before mailing itself to the next machine.

After performing these operations, the worm would sleep for a while. Then it would awaken and get into people's accounts by trying to guess their passwords. To do this, the worm tried combinations of the user name, the last, the first, last plus first, nicknames, and a list of about a thousand most commonly used passwords (e.g., aaa, arrow, banana, bob, cretin, dog, eager, finite, guntis, hello, heinlein, imperial, isis, japan, kermit, lambda, noxious, oracle, password, rabbit, smile, snoopy, tape, unicorn, virginia, weenie, yang, zap), all at a machine-gun rate of speed.

About 5 percent of the computers were accessed by breaking passwords. Most often, the worm got in via the bug in Fingerd and the back door of Sendmail. The
Virus!

worm's role in life was to reproduce, nothing more. Anytime the worm found a host, the worm tried to infect the host by one of the three methods described. The worm also occasionally sent bytes of information to a computer in the network at Berkeley called ERNIE. Apparently ERNIE was a monitoring mechanism by which the designer could track the progress of his creation through the Internet system.

Little damage was done, but suppose the KGB had been attacking us? And suppose the last instructions in the virus program were to destroy data, then delete itself? We might never have known what hit us.

Protecting unclassified computer networks is the responsibility of the National Institute of Standards in Technology, known until recently as the National Bureau of Standards. Defending military computer networks is the responsibility of the National Computer Security Center, an arm of the top-secret National Security Agency. The National Computer Security Center is given sole charge of safeguarding electronic secrets from foreign spies and vandals. And curiously enough, the Center's top scientist is the father of Robert Morris, Jr., Robert Morris, Sr.

And it was to his father that Robert Morris, Jr., turned that day after the virus attack. Robert Morris, Sr., fifty-six, himself a graduate of Harvard, worked for many years at AT&T's Bell Labs in New Jersey and, while there, developed a passion for the world of computer security.

Curiously enough, Morris, Sr., helped develop the UNIX operating system, a standard operating system for technical computing and the target of his son's worm. Two years ago, the elder Morris left Bell Labs to work as the chief scientist for the National Computer Security Center, plunging him into the "wilderness of mirrors" world of the intelligence community.

With top-secret clearance, Robert Morris, Sr., has an office with a specially secure Tempest computer terminal and blackboards covered with words in Russian, a language in which he is fluent.
It was five years ago that the elder Morris was a witness before a House committee studying a new and ominous phenomenon called the computer virus. He had bluntly stated, "The notion that we are raising a generation of children so technically sophisticated that they can outwit the best efforts of the security specialists of America's largest corporations and the military is utter nonsense. I wish it were true. That would bode well for the technological future of the country."

Ironically, it appears that it is his son, Robert Morris, Jr., who has so deftly outmaneuvered the high-security establishment. RTM, as he is known on the computer networks, is now the subject of inquiry by the FBI and U.S. attorneys in at least two states. Several of Morris, Jr.'s friends identified him as the creator of the virulent Internet virus, while his father is the man responsible for shielding Internet and other, more sensitive networks from such electronic intruders.

Robert Morris, Sr., is also the man who introduced his talented son to the science of computing and is torn by the furor surrounding the Internet incident. On the one hand, he condemns computer pranks as irresponsible, comparing such actions to teenagers "stealing a car for the purpose of joyriding." He has said that he believes the Internet incident "raises mostly a common sense question" about the responsible use of computers, and the instant fame created by his son's prank "is not a career plus."

On the other hand, Morris, Sr., has heard what other experts say of the Internet incident: that the worm was a programming triumph "fit for publication in a journal," that it caused no lasting damage, that it pointed up far more serious security threats. "I know a few dozen people in the country who could have done it," he says. "I could have done it, and I'm a darned good programmer."

Morris, Sr., and his son are regarded by their peers as brilliant computer analysts, blessed with insights into mathematical and logical problems whose complexity might confound lesser experts. Both began their profes-
sion in summer jobs at Bell Laboratories, and both main­tained computer systems while studying at Harvard Uni­versity. Both love the study of computer security and are intellectually entranced by the challenge of finding holes and "back doors" in supposedly secure computer pro­grams.

This strong alliance between father and son has led some industry watchers to question the innocence and intent of RTM's actions.

"It's important to understand the intelligence commu­nity mentality from the top level," says Wes Thomas, a well-known high-tech PR man and former intelligence officer. "In this community, viruses are clearly an expres­sion of an obsession for electronic warfare. This commu­nity perceives that we are heading toward viral war.

"There is a well-known intelligence process known as 'pulsing the system.' When flying planes in Vietnam, we would deliberately fly over enemy radar installations. Their radar would have to turn on to see who we were. As soon as that happened, we would grab the radar intelligence, and we would know what signals to jam. Then next time we flew over, we would jam those partic­ular frequencies or frequency strategies, thereby effec­tively blocking the enemy radar.

"It is my opinion that Robert Morris, Jr., pulsed the system to see what it would do. The intelligence commu­nity now has a tremendous amount of information about what happened. They have all kinds of data about viruses and worms, and even more importantly, how people reacted under emergency conditions. The human sociol­ogy of the Internet grid was exposed for the first time."

What evidence is there to suggest that RTM and his dad were part of an NSC plan to pulse the system? Ac­cording to Mr. Thomas, "AT&T fixed the trap door in the E-mail system long ago. And, curiously enough, Robert Morris, Jr., had worked on the E-mail system. He put in security precautions and the trap door of which he later took advantage. AT&T later fixed and blocked that trap door."
‘Now, why,’ one might ask, ‘if RTM knew the potential of backdoor abuse, and he was smart enough to fix it, and AT&T technicians knew how to fix it, don’t you think they would be a little more helpful to the rest of the community in fixing it, too? Isn’t there something weird happening here?’

‘By not alerting the rest of the UNIX community, by not saying, ‘Hey, you should take your debug out, there is a trap door there that can be opened,’ by not alerting the rest of the community to that trap door, aren’t they playing some serious games with us? Aren’t they pulsing the system?’

According to Mr. Thomas, this intelligence game is one of electronic countermeasures, of point, counterpoint designed to see how people handle the situation. It’s an adult version of the kid game where you get someone to chase you, and you watch how they do it.

So what is the real motive here? Mr. Thomas believes that the NSA wants to scare everybody because no one in the private sector has been listening to them.

‘Basically it’s a publicity gimmick to raise consciousness about computer security issues. The NSA believes that computer viruses and worms really are a significant risk to the health of our computers, and unless a lot of resources are allocated to the problem, we will be a victim of them. The government alone cannot solve the problem. Private corporations need to contribute resources as well. By pulsing the system, the NSA is telling us that there is a serious problem and that we need to do something about it.’

Interesting hypothesis, eh? It certainly moves RTM’s motivation for creating the Internet worm virus into a new light. What does the actual evidence suggest? Let’s take a look.

The worm did take advantage of an open back door into the system. Who left the door open and why?

Eric Allman, a senior systems programmer at the International Computer Science Institute in Berkeley, California, is the actual creator of the Sendmail program
through which the virus entered the Internet system. He developed Sendmail during the Ingres project at the University of California at Berkeley in 1978 through 1979. Allman was never able to perfect the Sendmail system, because it was appropriated for use before he was finished. "I wasn't allowed to fix the bugs on the system," he said. So he put a backdoor debugging system in the network to enter whenever he wanted.

"When you turned on the debugging system, it disabled a security feature that keeps a user from sending mail [if the system wasn't accessed properly]. I figured that if you were already in the system, you didn't need the security feature," Allman said.

So how could the Robert Morrises have learned about the back door?

RTM worked for a time at AT&T's Bell Laboratories. In 1987 Bell Lab techs closed the secret back door in Sendmail. Douglas McIlroy, a member of the technical staff, said Bell tried to warn other institutions of the potential for a breach in security, but no one listened.

Was the massive replication of the worm truly a mistake?

Friends say RTM originally intended for a single copy of the rogue program to exist on each machine in the Internet network. If this is so, then revealing the existence of such a secret widespread infestation would have had even more impact than what has already transpired. It is reminiscent of the bug-infested U.S. embassy scenario in Moscow.

Let's look at some other evidence. How are the federal authorities treating RTM? If Mr. Thomas is right, they should be treating him pretty well.

Within a week of the outbreak, Robert Morris, Sr.'s, employer, the National Computer Security Center, organized a meeting of computer security experts with representatives of twelve federal agencies and announced that RTM's program had "no dangerous hidden features."

"There was no sign in the code that it intended to do anything but spread around the country," claimed Mark
Eichen, an MIT programmer who disassembled the program.

They are basically saying it was a first-rate hack, and no real damage was caused. As a matter of fact, a lot of people learned more about real issues of computer security in a day by being on the front lines of the attack than they would have in months in the classroom studying the topic. So they are suggesting we give the designer some slack.

The NCSC meeting produced a number of proposals, including one that the National Security Agency and the National Institute for Standards in Technology create a clearinghouse for information about virus-type programs. Another proposal called for the creation of an emergency "swat" team to fight such programs. This fits with Mr. Thomas's contentions that the government wants massive public support in fighting this high-tech problem.

It is also interesting to note that the Washington lawyer retained by RTM, Thomas A. Guidoboni, decided that his client would not talk to the FBI until a couple of weeks after the incident. And the FBI, conducting a full-scale criminal investigation of the incident, went along with that decision. One might ask what kind of pressure it takes to keep the FBI so well in line.

In its investigation, the FBI claims that it is examining possible violations of the Computer Fraud and Abuse Act of 1986, a law under which there have been no prosecutions to date. The law makes it a misdemeanor, punishable by a fine and up to a year in jail, to knowingly use a government computer without authorization and to disrupt its normal operation. A second violation is a felony and can be punished by a fine and up to ten years in jail.

Curiously, the FBI considered and then rejected the idea of seeking grand jury subpoenas for documents at Cornell that would shed some light on the incident. Why would the Bureau ignore data implicating RTM's role in the design and implementation of the virus? Perhaps because the NSA has already achieved its first unstated
goal—to focus widespread attention on the virus issue. Why hurt the kid who has helped the agency so much? And you can bet that with all this publicity about the need for NSA services, their budget just got a shot in the arm.

Let's look a little more closely at the Robert Morrises. Physical appearances alone suggest Robert Morris, Sr., is a pretty radical guy and that the power he wields within government and industry allows him to live very well, dress the way he wants, command lots of people, and spend a lot of money on what he thinks is important. Having played a major role in the design and implementation of the UNIX operating system, he is more than simply a supporter of the system. He is the system. UNIX runs almost sixty thousand government computers, containing, among other things, most of our nation's vital defense secrets.

He is better than good, according to his lifelong associates from Bell Labs. His programs are the rule books by which computers run. "He's one of the best," said his one-time supervisor at Bell Labs, M. Douglas McIlroy.

And you can bet that most, if not all, of the "leaders" of our country, including the generals, CEOs, and elected representatives, know diddly-squat about computers, compared to Robert Morris, Sr. He could single-handedly topple our government, if he felt so inclined.

He met his wife, Anne, in 1959, while she was working at MIT and he was pursuing a doctoral degree in mathematics at Harvard. A summer job at Bell Labs caused him to abandon his pursuit of a Ph.D. in favor of running a range of research projects at Bell. Robert spent twenty-six years at Bell Labs and even had a remote terminal in his two-hundred-year-old family farmhouse during the 1960s.

He and Anne had three sons, Meredith, Robert, and Ben. Robert was the only one really interested in computers and technology. At the age of four, he was using cardboard and Scotch tape to construct intricate models, including an automobile that could be steered. At nine,
he was reading old issues of *Scientific American*. “We used to buy them at old book sales,” says Anne. “We still have them—steamer trunks full—in the attic.”

By the time young Robert was fourteen, his “sleuthing from the outside” on the family computer terminal attracted attention from Bell officials, who began to allow him to visit his dad’s office and play with computer projects. He wrote and analyzed programs and became quite the hacker, sneaking in and out of computer files just for fun.

“I never told myself that there was anything wrong with what I was doing,” he was quoted as saying in a 1982 magazine article, adding that he continued with it because he enjoyed the challenge of testing computer security.

Understandable, considering that his dad was a recognized expert in code breaking and electronic security. Robert Morris, Sr., was so good in these matters that Bell Labs’ parent, AT&T, gave him special authority to attempt break-ins into Bell computers so that system “holes” could be uncovered.

After a couple of summer jobs at Bell Labs, Robert Morris, Jr., entered Harvard and worked part-time for the University’s Aiken Computer Center as a maintenance worker. “He already had quite a reputation as a UNIX expert,” said a college roommate. “People were very impressed with him.”

At Harvard, he developed the reputation of being a “gentle prankster.” On one occasion, he created a program that swept computer users into an electronic maze whenever they committed a certain misspelling. Another program caused users’ portraits to bounce around the computer screen in idle moments. Once he rearranged the furniture in the school library. Another time he shined a low-level laser onto a Stop sign at night, causing motorists to wonder if they were hallucinating. Never really destructive, he obviously was simply having fun with his pranks. His maintenance ability gave him “super-user” privileges at the Aiken Center, allowing him
complete access to high-security files in the system. Yet he never abused them.

"It was no challenge," claimed his former roommate. "He was already inside. Why do anything?"

So, keeping the mentalities of the Morrices in mind, doesn't it seem possible that they may have combined their awesome computing talents to create a serious warning to our nation on our cybervulnerabilities?

Note: A more technical analysis and description of the Internet Worm virus can be found in Appendix A.
Once upon a time, a disgruntled mainframe programmer was fired by the administrator overlords and summarily removed from the computer sanctum. All was well for six months, six days, six hours.

Suddenly, all the keyboards on the mainframe's terminals mysteriously ceased to function as the programmer's personally planted time bomb proceeded to lobotomize the system.

The administrators watched in horror as the tape drives locked up and all mounted tapes were erased, bit by bit. There was absolutely nothing they could do as the card reader/punch proceeded to punch holes randomly in all program decks left in the card hopper.

Finally, the disk and drum storage devices went through a complete erasing process, sending all their data to the Data Bardo. Meanwhile, the time bomb dutifully displayed its moment-by-moment blows on the main console monitor.

Fortunately, the great sanctum had recently made a
backup of all its data. At great expense to the administrators, the sanctum programmers spent weeks restoring and generally recreating all lost files.

Then a special team of crack programmers were hired to comb the operating system's source code carefully in search of the deadly time bomb.

Finally, they found it and, with the skill of practiced surgeons, removed all traces of the software cancer. Once rebooted, the sanctum's system behaved beautifully, without a hitch.

That is, until six months, six days, six hours later, when the whole process repeated itself.

This is no fairy tale. This story is based on an actual incident that occurred in the late 1960s, a time before personal computers, when giant dinosaurlike mainframes roamed the planet.
Let us begin our story with the first known sighting of what are now known as computer viruses.

Legends From the Cyberpast

From the primordial soup, there arose an assembly language program called Creeper. A very simple creature—its only function in life was to perpetuate itself, duplicating every time it was run. So that it could clone more efficiently, Creeper's creator/programmer gave it the ability to spread from one computer to another within the constellation of its home network.

Before long, copies of Creeper were everywhere, crowding out more useful programs and data. The network programmers considered it a growing infestation out of control. The programmers thought of fighting fire with fire.

They created a second self-replicating program called Reaper. Reaper's purpose was to seek out and destroy copies of Creeper. Reaper, too, performed its task admirably, traveling from computer to computer, finding copies of Creeper and removing them from existence. And when the last Creeper was deleted from the system, Reaper destroyed itself.

The first known virus, the Creeper, was first sighted in 1970. Designed by Bob Thomas of BBN (Bulletin Board Network), the Creeper was a demonstration program that got out of control. It crawled through ARPAnet, the nationwide Pentagon-funded network linking university, military, and corporate computers. It left a message wherever it went, saying, "I'm the Creeper, catch me if you can!"

The Creeper created relatively few problems, other than taking up a lot of space on the computer networks. Later evolutions of the computer virus were not nearly so kind.
CORE WAR: PLAYING WITH PANDORA'S BOX

The venerable computer legend of Creeper and Reaper inspired Professor A. K. Dewdney, at the University of Western Ontario, to help popularize a new and highly innovative computer game, Core War.

The term Core War originated in outmoded memory technology. Before semiconductors were used for memory storage, a computer's memory was built out of thousands of ferromagnetic cores, or rings, strung on a mesh of fine wires. Each core retained one bit, the fundamental unit of digital information.

Though modern semiconductor and CD technology are used today for memory storage, the active part of the memory system, where a program is kept while it is being executed, is still referred to as “core memory” or “core.”

Professor Dewdney, a highly learned, inventive soul, took great pleasure in allowing his readers and students to participate in a game that was interactive, creative, and great fun to boot. But what raised Professor Dewdney to the ranks of a true gamesmaster was his willingness to allow Core War players to taste the pleasure of an ancient human ritual—the pleasure of the hunt.

First, publicly described in Professor Dewdney's "Computer Recreations" column in the May 1984 issue of Scientific American, Core War is a game where two computer programs stalk each other in the domain of their native habitat—the memory chips of a digital computer.

Like two enemy commandos battling each other to the death on a South Pacific island, the players in Core War are adversaries in a contest of which the outcome depends on how well prepared and capable the battling programs are. These commandos have special capabilities, including the ability to replicate themselves instantly.

The first actual Core War tournament took place in 1984 at the historical Boston Computer Museum, a
unique environment where shiny new ultrafast microcomputers sit in the shadows of their venerable roomsized, air-conditioned ancestors, forever changing the definition of computer gamesmanship.

The idea of computer programs destroying each other became a popular high-tech pastime. Many of the techniques developed in fun while playing Core War would later become cornerstones of the new computer virus technology.

**Of Dwarfs and Imps**

Core War's first warrior battle program was called Dwarf. It was stupid, yet very dangerous. Like the knight in a game of chess, Dwarf sends instructions that skip over addresses in the memory array, bombarding every fifth address with a zero.

A zero signifies a nonexecutable data statement. A zero dropped into the middle of an enemy program can bring it to a halt. Dwarf itself is immobile, but its artillery threatens the entire memory array.

No enemy longer than four instructions could avoid taking a direct hit from Dwarf's artillery. To oppose Dwarf, you, as creator of the opponent program, had only three options:

1. **Mobility**—Your program could move about, thereby eluding a direct hit.

2. **Defense**—Your program could absorb hits from Dwarf and repair the damage.

3. **Offense**—Your program could use its weapons to destroy Dwarf first.

Offensive measures against Dwarf were difficult. You had no idea exactly where Dwarf was located in the memory array, and it usually took at least 1,600 execution cycles (turns to play) before Dwarf was hit.

In Core War combats pitting a Dwarf against another
Dwarf, each program won 30 percent of the time. In 40 percent of the contests, neither program scored a fatal hit.

Exercising the mobility option, Dewdney designed a one-line battle program called Imp.

Like a slinky snail leaving a trail, Imp pours itself from its present location into the next adjacent memory address. As the program is executed, Imp moves through the array at a speed of one address per cycle, leaving behind a trail of "MOV 01" instructions.

What happened when an Imp was pitted against a Dwarf? Like machine-gun fire, Dwarf's zero barrage moved through the array much faster than Imp could move. However, because Imp was only a single address in size, Dwarf's barrage might never hit the tiny Imp.

If Imp reached Dwarf first, Imp would plow through Dwarf's programming code, effectively subverting Dwarf. Interestingly enough, a direct hit to a specific location of Dwarf's code turned it into a second Imp, which endlessly chased the first one around the memory array. In Core War, such an outcome was considered a draw.

Imp and Dwarf represent a class of battle programs now considered small and aggressive, but not intelligent. The second-generation programs were larger and somewhat less aggressive, but smart enough to deal with the programs in the lower class.

Next of Kin

The Gemini program was the core of this smarter generation. Gemini was not a complete battle program but a mobility mechanism whose only function was to make a copy of itself 100 addresses beyond its present position and then transfer execution to the new copy.

Two examples of new battle programs using the Gemini mobility mechanism were Juggernaut and Bigfoot. Juggernaut copied itself ten locations ahead instead of 100, overwriting its opposition (as did Imp). Bigfoot,
however, used as its interval between copies a large prime number. Bigfoot was harder to catch and had the same destructive effect on enemy code as Juggernaut.

Both won far more often than Imp, leading to fewer draws because an overwritten program was less likely to be able to execute fragments of its code.

**Raidar, Scanner, and Picket Fences**

Examples of the third-generation program were Raidar and Scanner.

Raidar employed a simple yet effective early-warning enemy-detection system. It moved out of the way when potential danger approached. The detection system consisted of two “picket fences” surrounding the program itself. Each picket consisted of 100 consecutive addresses filled with ones and was separated from the program by a buffer zone of 100 empty addresses.

Employing a two-part offensive system, Raidar divided its time between systematically attacking distant areas of the memory array and checking the status of its picket addresses. If a picket was altered, Raidar interpreted it as an attack by a Dwarf- or Imp-class program. Raidar then copied itself to the other side of the damaged picket, constructed new pickets on its unprotected side, and resumed normal operation.

Scanner, created by Professor Dewdney’s student assistant, David Jones, was a self-repairing program that could survive some attacks. It did this by maintaining two copies of itself, executing one of them, and keeping a redundant twin for comparison and repair.

The active copy periodically scanned the twin to see if any of its instructions had been altered by an attack. Changes were detected by comparing the two copies, with the assumption that the executing copy was correct. If any bad instructions were found, they were replaced, and control was immediately transferred to the twin, which then began to scan the first one.
Redcode Is a Fightin' Word!

As an educator, inventor, and journalist, Professor Dewdney encouraged the excited young fans of Core War to experiment, experiment, experiment. He suggested, for example, the self-repairing, self-monitoring mechanism used in Scanner.

He helped establish (and often updated) the Core War memory array game board and developed Redcode, a particularly feisty assembly-type programming language designed specifically for constructing battle programs.

Professor Dewdney was also responsible for developing MARS (Memory Array Redcode Simulator), an executive program that functioned like an overseer of the battle programs. Performing much as a simple time-sharing system, MARS permitted the battle programs to take turns—first one side, then the other.

With MARS, you could watch the action and otherwise monitor the progress of the game as Dwarfs fired zeros at passing Imps, Raidar programs monitored their pickets, and Scanners nervously prepared themselves for self-repair and duplication.

Many a case of soda pop was consumed while Core War programmers glared expectantly into the green phosphorus of the game screen as their programs did battle, awaiting the emergence of the lone victor.

Professor Dewdney's fundamental motivation was to create an exciting and stimulating game: "Only if reasonably robust programs can be developed," he noted, "will Core War reach the level of an exciting game, where the emphasis is shifted from defense to offense. Battle programs will then have to seek out and identify enemy code and mount an intensive attack wherever it is found."

Escape from the Core

Dewdney was overwhelmed by the response he received from his 1984 Scientific American column. He had developed Core War as a game; it had not occurred to him just how serious a topic he had raised.
What happens if these battle programs escape from their memory array game board and into the real computer world? What if they are assisted by human programmers who help adapt and familiarize the battle programs with the new terrain?

To get a sense of the possibility, imagine yourself pitted against a fully equipped combat-ready warrior who is half-man, half-machine and roams the streets, systematically terminating the life functions of every fifth person within a square block. Such is the experience of your Lotus 1-2-3 or dBASE III program when confronted with a Dwarf-class battle program.

"Some of the possibilities are so horrifying that I hesitate to set them down at all," Dewdney wrote in his March 1985 *Scientific American* column. "I am tempted to say that if we must have war, by all means let it be soft. On the other hand, the possibility of an accident due to the intimate connection between military software and weapons-control systems gives me pause."

He recounted the plot of a newly published French spy thriller, *Softwar: La Guerre Douce*, about the purchase by the Soviet Union of an American supercomputer. Authors Breton and Beneich spin a diabolical yarn about American plans to secretly infect the machine with a "software bomb." Supposedly purchased to help with weather forecasting in the Soviet Union, the software contained a hidden trigger; as soon as the U.S. National Weather Service reported a certain temperature at St. Thomas in the Virgin Islands the program would proceed to subvert and destroy every piece of software it could find in the Soviet computer network.

Or even more horrifying still, consider a massive Strategic Defense Initiative (SDI) system, containing several million lines of code and miles of telecommunication networks, controlling space-orbiting laser cannons and nuclear devices. Imagine it riddled with viruses—Imps, Dwarfs, and who knows what other unknown, near-invisible creatures—a system rendered unusable at a crucial moment in our nation's fragile future.
Soon chilling real-life stories poured into Professor Dewdney's office.

It's Alive, Professor! It's Alive!

Among the hundreds of letters received by Professor Dewdney, the most terrifying was a letter from Italy written by Marco and Roberto, two Core War fans and Apple II hackers, who had read a translation of his column. The correspondence from the two Italians noted:

"Marco thought to write a program capable of passing from one computer to the other, like a virus, and 'infecting' in this way other Apples. But we were not able to conceive it until I realized that the program had to 'infect' floppy disks, and use the computers only like a media from one disk to the other. As you know, every Apple diskette contains a copy of the Disk Operating System, which is bootstrapped by the computer at power on.

"The virus," the letter continued, "was an alteration in this DOS, which at every write operation checked its presence on the disk, and, if not [present], would modify in the same way the DOS on the disk, thus copying itself on every diskette put in the drive after the first power up.

"To make the virus malignant, we decided that after 16 self-reproduction cycles, counted on the disk itself, the program would decide to re-initialize the disk immediately after bootstrap [erasing all data on the disk!].

"We thought that installing such a DOS on some disks used in the biggest computer shop in our city, Brescia, would cause an epidemic to spread in the city."

Fortunately, Marco and Roberto never tried to test their idea.

Someone who did, however, was Richard Skrenta, Jr., a high school student in the steel mill town of Pittsburgh, Pennsylvania, and a Core War fan. His infectious design caused subtle errors to appear throughout the operating system of his Apple II personal computer.
"All of this seems pretty juvenile now," reflected Skrenta in the winter of 1985, "but, oh, woe to me! I have never been able to get rid of my electronic plague: It infested all of my disks and all of my friends’ disks."

Feeling obligated to abate the plague he had launched upon himself and his friends, Skrenta eventually did devise a program to destroy the virus, but alas, it was never as effective as the original virus itself. As someone once said, "It’s not nice to mess with Mother DOS!"

The most significant aspect of Core War is that it taught thousands of bright young hacker minds how to do battle within a computer’s memory. This knowledge would later be used as the groundwork for designing non-game-oriented destructive programs, such as the viruses currently plaguing our computer networks and disks today.

MEANWHILE, ON A CAMPUS NOT FAR AWAY

Evolutionary computer genius Dr. Fred Cohen performs major eye scrunching before a terminal connected to a heavily loaded VAX 11/750 minicomputer.

Dr. Cohen has been jamming eight hours straight, readying the first truly replicating computer virus for demonstration before a very special gathering of security-focused individuals. He is about to perform five viral experiments before the group, demonstrating the efficacy of computer viruses as a means of taking over a computer’s operating system.

His goal: to make the world’s first truly replicating virus that can infect and take complete control over a whole computer system . . . and do it fast!

His quest: to create the ultimate virus. People who pursue Core War ask themselves: "Can I create a better program than you can?" Dr. Cohen asks: "What is the best program I can ever find?"
Dr. Fred Cohen is a recognized expert in computer systems and security, currently teaching at the University of Cincinnati. An inquisitive, insightful, and quick-witted man, Dr. Cohen had a major programming breakthrough while attending the University of Southern California’s electrical engineering doctorate program in the fall of 1983. In a recent interview, Dr. Cohen described his breakthrough experience.

"I have been interested in various attacks on computers since I was ten or eleven," recounted Dr. Cohen. "And I’ve known many people in the various communities that have associated with this from time to time.

"This whole thing came to a head," he said, "while I was sitting in a class taught by cryptographer Len Adleman at USC. The concept of the virus was something that just appeared to me in that class. It was like the proverbial light bulb turning on above the head in the cartoon. All of a sudden, I realized that this thing was possible and that the whole world of computers as we know it was over! It was all over in a moment."

Dr. Cohen continued, "As far as I could tell at that moment, you could forget about every computer system in the world. Once you understand the principle and you have been working in computer security for a long time, it is so obvious."

 Shortly after his viral insight, Dr. Cohen arranged a demonstration of his evolutionary breakthrough at the weekly computer and security gathering at the university.

A few days before the session, Dr. Cohen composed the first truly replicating computer virus in only eight hours of programming time, using mostly assembly language instructions.

With eager anticipation Dr. Cohen’s peer group of computer and security experts gathered around the chosen VAX 11/750 minicomputer, and the experiment began.
In launching the system-takeover experiment, Cohen implanted the initial infection in VD, a directory program that would help track the virus as it replicated its way through the VAX's UNIX operating system.

The virus was cleverly implanted at the beginning of the program so that it was executed before any other system processing could occur, and Dr. Cohen took extraordinary precautions to keep the virus under control while it infected. Each infection was performed manually, and no actual damage was done, only the silent reporting of the virus's progress into history.

Dr. Cohen used tracers to make sure the virus would not spread without detection. In addition, access controls were used for the infection process, code required for the attack was kept in segments, each encrypted and protected to prevent illicit use.

The results of these carefully controlled experiments were extraordinary. "In each of five attacks, all system rights were granted to the attacker in under an hour," Dr. Cohen reported. "The shortest time was under five minutes, and the average under thirty minutes. Even those who knew the attack was taking place were infected."

Dr. Cohen had expected the attack to be successful, but to take over the highest levels of security within the system in such a short period of time was absolutely shocking. The average user level, the administrative level, and the highest security accesses to the system were all easily penetrated. The virus was so fast (under half a second) that the delay caused by the infection process went unnoticed by the observers!

Cohen was excited by his results, seeing all kinds of possibilities for his new creation, and quickly planned even more ambitious experiments.

At first, he did not even have a name for his new creation. But, as Cohen's creation replicated and penetrated level after level of system security without detection, his friend and cryptographic colleague, Dr. Len Adleman, was reminded of a dreaded living pattern previously seen only in nature—viruses.
Dr. Cohen's elation at his success, however, was quickly squelched by the human administrative powers watchdogging his work. Once the results of his experiments had been announced, the administrators decided that no further computer security experiments would be permitted on their expensive computer systems. The campus administration's ban was across the board, including Cohen's plans for immediate development of tracers that track potential viruses and new password systems that could improve system security.

"This apparent fear reaction is typical," says Cohen. "It is hard to take when somebody tells you that your own world isn't worth anything anymore. That's the experience I had with these administrators. They felt raped.

"When you are on a computer system you have been using for years, and you have your entire book on the system, you trust that system. If it all disappears tomorrow or the day before it's supposed to be done, you will feel raped. You will feel like a victim. That's been the experience. You have to learn, in the words of the rape crisis centers, to be a survivor, not a victim. Well, that's life! I like that phrase, because that is what we are talking about."

A Man Without a System

After these successful experiments in the UNIX environment, Dr. Cohen felt convinced that the same viral techniques would work on many other computer operating systems. All he had to do was find someone willing to lend their expensive systems to him for further experimentation and testing. This proved to be quite a challenge, for who wanted an infected system?

Dr. Cohen's first attempts to test the virulence of the viruses involved further negotiation with more fearful administrators. While negotiating, Dr. Cohen designed experiments for a host of minicomputers that might soon become available: a Tops-20 system, a VMS system, a VM/370 system, and any connecting network.
To help convince the administrators of his sincerity, Cohen developed prototype viruses for each of the available systems and demonstrated the viruses' ability to find files to be infected, infect them, and cross user boundaries seemingly at will.

The powers that be were not impressed. According to Dr. Cohen, the security officer at the facility was in constant opposition to these experiments and would not even read the good doctor's proposals!

Dr. Cohen had even offered security officers and system programmers the opportunity to observe and oversee every aspect of all experiments. But to no avail. The system administrators were simply unwilling to allow any form of official research, including even off-line analysis of viral threats using sanitized log tapes! Once again, the viral doctor was experiencing the all too human reaction of fear of the unknown and its companion reaction—personal rejection.

The Univac 1108 Virus

Finally, in July 1984, Dr. Cohen got a break and for two weeks was able to gain part-time access to an ancient Univac 1108. Because Cohen was pressed for time and unfamiliar with the programming environment of the Univac, many minor elements were ignored in the design and implementation of this next wave of viral experiments.

With the assistance of a more experienced Univac programmer, Dr. Cohen designed the Univac 1108 virus to infect only one program at a time, allowing its progress to be clearly demonstrated without involving a large number of users or programs.

As a security precaution, the system was isolated, used with only a system disk, one terminal, one printer, and dummy accounts created solely for these experiments.

Dr. Cohen's viral code consisted of five lines of assembly code, about two hundred lines of Fortran code, and about fifty lines of command files. After about eighteen
hours of programming time, the virus was ready to perform its first infection.

What went down in cyberhistory as the "1108 virus" was then demonstrated to a group of about ten of Dr. Cohen's favorite people, who included key administrators, programmers, and security officers. Once again, Cohen dazzled the group with a performance that clearly demonstrated a virus's ability to cross user boundaries and effortlessly move up security levels in a computer system.

Dr. Cohen estimated at the time that a competent systems programmer could write a much better virus for the system in under two weeks. He also found that once a programmer grasped the method of attack, developing a specific virus was not difficult. In fact, all of the programmers involved in the experiment believed that they could build a better virus faster the next time around.

Once again, Lady Luck turned in the doctor's direction, and in August 1984, Cohen was given permission to equip a VAX UNIX system for detailed research in analyzing the spread of viruses.

**Virulence of the Viruses**

Through this next round of seminal viral research, Dr. Cohen made several interesting discoveries:

- The degree of program sharing by users varied greatly from system to system with no obvious reason why this was the case.

- A small number of users appear to account for the vast majority of program sharing. These people dramatically speed up the process of system infection. A virus could be greatly slowed by protecting these program sharers with antiviral software measures.

- Many of these major sharers turned out to be system administrators, and if any of them were infected, the entire system would likely fall within minutes. This is because UNIX is designed to prevent the lower-level users from accessing higher-level administrative accounts, but not designed to prevent access from the
top down. Once the high-level accounts get infected, the virus spreads quickly down through the whole network!

Cohen’s detailed analysis showed that system administrators tended to try new programs on the system bulletin board as soon as they were announced, producing almost instant networkwide infection—a classic case of curiosity killing the cat.

“This allowed normal users to infect system files within minutes,” Dr. Cohen explains. “Administrators used their accounts for running other users’ programs and storing commonly executed system files, and several normal users owned very commonly used files,” he continued. “These conditions make viral attack very quick. The use of separate accounts for systems administrators during normal use was immediately suggested, and the systematic movement of commonly used programs into the system domain was also considered.”

**Cohen’s Conclusions**

<table>
<thead>
<tr>
<th>UNIX C</th>
<th>Bell-LaPadula</th>
<th>Instrumentation</th>
<th>Shell</th>
<th>VMS</th>
<th>Basic</th>
<th>DOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>8 hr</td>
<td>18 hr</td>
<td>N/A</td>
<td>15 min</td>
<td>30 min</td>
<td>2 hr</td>
</tr>
<tr>
<td>Inf. Time</td>
<td>5 sec</td>
<td>20 sec</td>
<td>N/A</td>
<td>2 sec</td>
<td>2 sec</td>
<td>15 sec</td>
</tr>
<tr>
<td>Code</td>
<td>200 lines</td>
<td>260 lines</td>
<td>N/A</td>
<td>7 lines</td>
<td>9 lines</td>
<td>30 lines</td>
</tr>
<tr>
<td>Trials</td>
<td>5</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Min. Time</td>
<td>5 min</td>
<td>N/A</td>
<td>30 sec</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Average Time</td>
<td>30 min</td>
<td>N/A</td>
<td>30 min</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Max. Time</td>
<td>60 min</td>
<td>N/A</td>
<td>48 hr</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

From these early experiments, Dr. Cohen concludes, “Viral attacks appear to be easy to develop in a very short time, can be designed to leave few if any traces in most
current systems, are effective against modern security policies for multilevel usage, and require only minimal expertise to implement.

"Their potential threat is severe, and they can spread very quickly through a computer system," Dr. Cohen continues. "It appears that they can spread through computer networks in the same way as they spread through computers, and thus present a widespread and fairly immediate threat to many current systems."

What was particularly disturbing were his comments on supposedly secure computer networks: "Nearly every 'secure' system currently under development is based on the Bell-LaPadula or lattice policy alone, and this work has clearly demonstrated that these models are insufficient to prevent viral attack."

When questioned about his experiences in dealing with university bureaucracies, Dr. Cohen had to put in his insightful licks to the administrators.

"The problems with policies that prevent controlled security experiments are clear. Denying users the ability to continue their work promotes illicit attacks, and if one user can launch an attack without using system bugs or special knowledge, other users will also be able to do so.

"By simply telling users not to launch attacks, little is accomplished. Users who can be trusted will not launch attacks, but users who would do damage cannot be trusted, so only legitimate work is blocked. The perspective that every attack allowed to take place reduces security is, in my opinion, a fallacy. . . .

"The idea of using attacks to learn of problems is even required by government policies for trusted [i.e., national security-level] systems," Cohen points out. "It would be more rational to use open and controlled experiments as a resource to improve security."

Swiss-Cheese Security

"To be perfectly secure against viral attacks," Dr. Cohen warns, "a system must protect against incoming
information flow, while to be secure against leakage of information, a system must protect against outgoing information flow.

"But in order for systems to allow sharing, there must be some information flow. It is therefore my major conclusion that the goals of sharing in a general-purpose, multilevel security system may be in such direct opposition to the goals of viral security as to make their reconciliation and coexistence impossible."

It appears that as long as we share computer resources, facilities, and software—be they networks or floppy disks—we will be exposed to viral attack! Comforting, eh?

So what do we find the good doctor thinking about currently?

"The most important ongoing research involves the effect of viruses on computer networks," says Dr. Cohen. "Of primary interest is determining how quickly a virus could spread to a large percentage of the computers of the world."

One of Dr. Cohen's most valuable contributions to viral technology was the development of the replication process. This process is similar to the delivery component of an intercontinental ballistic missile. Viruses are composed of two distinct parts, the delivery system and a payload, or warhead. Without the warhead, a virus cannot cause much damage. But with a warhead in place, the destructive capacity and, hence, the stakes are much, much higher.

It's lucky for us that Dr. Cohen is on our side. For as Professor A. K. Dewdney told me, "If Fred ever had a psychotic episode or something, it would be a terrifying thing."
The Legend of the Cookie Monster

Once upon a time, there was a big computer system called DECsystem 10, one of the most powerful of its type in the land. Many of the programmers of this system were aware of a strange creature that lurked within.

While not harmful, this creature learned about human eating habits from the programmers. It liked to secretly lie in wait for its favorite humans to log on to the system, and then it would pop up on the programmer's screen with the message:

I WANT A COOKIE!

Only after the human typed in "COOKIE" would the system resume processing, and the cookie monster would go to sleep for a few days or so. Only those programmers truly in the know were in possession of the secret response that would send the cookie monster to sleep for weeks on end:

OREO!
The classic cookie monster was a special creature in that it demanded an antidote, an appeasement gesture, before it was happy. This kind of scenario has recently surfaced again with the first microcomputer virus, the Pakistani Brain.

**THE STRANGE BRAIN VIRUS OF THE PAKISTANI BROTHERS**

The year 1985 came and went without any serious viral invasion. Dr. Cohen's warnings of worldwide disaster were quietly put aside by the computer security community in favor of other issues that seemed more real: unauthorized system access, software piracy, and the design of new schemes for software copy protection.

In March 1985, Professor Dewdney was just receiving his chilling correspondence from the players of Core War. Even though the potential for disaster was growing, the security community still perceived viruses as the stuff of science fiction, and no action would be taken until a real disaster actually occurred.

We did not have to wait long. Science fiction was rapidly becoming real behind the closed doors of secret research facilities around the country. Reminiscent of biological warfare research, the famed Bell Laboratories and the National Security Agency were both conducting classified experiments, exploring the possibilities of widespread infection of viruses. More than a few viral experiments were being performed by late-night college hackers around the country.

With all this activity, you might have expected that the planet's first influential virus would have originated from someplace like California's Silicon Valley, the deepest interior of IBM, or at least an American junior high school. After all, America is the most innovative high-tech spot on the planet. Right?

But, with over forty million computers in worldwide use, there are now at least one million people capable of
creating a virus. So perhaps it is not so surprising that one of the first viruses to infect the world of the IBM PC came from a small company store located in Lahore, Pakistan.

A seemingly insignificant operation run by two brothers, Basit Farooq Alvi, nineteen, and Amjad Farooq Alvi, twenty-six, Brain Computer Services sold IBM PC software to local merchants and tourists at cut-rate prices. The brothers were selling brand-name megabuck computer programs from their carpeted storefront shop, programs like WordStar and Lotus 1-2-3 for as little as $1.50 a copy.

The brothers represented a new generation of Pakistanis—educated, worldly wise, and computer literate.

The proprietor of the computer store, Amjad, graduated from Punjab University with a degree in physics. He and his brother then taught themselves the basics of computer repair and programming in the IBM PC environment.
In 1985, while the brothers were producing customized software for businesses in Lahore, they were surprised to find that their software was being pirated by other businesses in Lahore and used without their knowledge or permission.

In January 1986, Amjad decided to do something about this wanton pirating, so he came up with the idea of creating a virus that would infect an unauthorized user’s computer, disrupt operations, and force him to contact Amjad for repairs. He designed the self-replicating virus and installed it in his customized software.

Meanwhile, the Alvi brothers were duplicating expensive American programs like Lotus 1-2-3 and WordStar, and injecting the virus into these programs as well. They then sold these infected programs to the visiting foreigners frequenting their Lahore computer store. The foreigners were mostly budget-minded American students and backpackers on vacation. They bought the cheap, tainted software and returned with it to the United States.

Every time they lent their new disk to a friend, and every time the disk was run on a machine shared by other users, the infected code spread from one computer to another. The first widescale computer virus epidemic was under way.

**Incident at Providence**

Froma Joselow, a financial reporter at the Providence Journal-Bulletin in Rhode Island, carefully slipped her IBM diskette with six months of notes and interviews into one of the newsroom PCs. Instead of her notes appearing on the screen, the dire message “DISK ERROR” flashed before her eyes!

“I got that sinking feeling,” she told Scott Brown, a Time magazine reporter investigating the incident. “Every writing project of mine was on that disk.”

Joselow took her failing floppy to the newspaper’s computer expert, systems engineer Peter Scheidler. Us-
ing special editing software, akin to a powerful magnify­
ing glass, Scheidler examined the disk's contents line by
line. "What I saw wasn't pretty," he noted. "It was gar­
bage, a real mess."

In an attempt to salvage Joselow's work, he peered into
each of the diskette's 360 concentric rings of data. As he
examined the boot sector of the diskette, a chill traveled
through his body. What he found was evidence of inten­
tional sabotage, evidence linking Joselow's disaster with
the brothers in Pakistan. Right there in the boot sector
was the following message:

WELCOME TO THE DUNGEON.
BEWARE OF THIS VIRUS.
CONTACT US FOR VACCINATION.

The message included the address and phone number
of Brain Computer Services in Lahore, Pakistan, and the
names of the two brothers. Good detective work on
Scheidler's part, but not soon enough to prevent many of
the hard disks at the newspaper from becoming infected.

When the reporters at the Providence Journal-Bulletin
dialed the Pakistani phone number listed in the code, Basit Alvi answered and said that they were the first
people to call in for the cure. He also expressed surprise
that the virus had traveled so far and refused to give his
last name over the phone.

"The antidote," he said, "is to substitute a clean oper­
at ing system for the contaminated one. I believe in the
power of substitution!" Alvi also claimed he wrote the
virus not to destroy data but as a warning to would-be
software pirates. The motive was to "punish" computer
users for buying and selling bootleg software, depriving
merchants of potential sales.

Like vigilantes of the old West, the brothers had taken
it upon themselves to personally punish software
thieves. For them, creating a homemade virus to punish
copiers seemed like a good idea at the time. "Because
you are pirating," says Basit Alvi, "you must be pun­
ished."
Unfortunately, the virus got out of control, spreading through the United States like a forest fire in a high wind. The Pakistani Brain virus infected an estimated 100,000 IBM PC disks across the United States, including some 10,000 at George Washington University alone. And, like any forest fire, the virus attracted fire fighters.

One of the first professional virus fighters on the scene was John McAfee, president of a new antiviral company, Interpath Corporation in Sunnyvale, California. A voice synthesizer engineer and a meticulously dressed man with precise speech patterns, he discusses computer viruses with the authority of a medical doctor describing the infection process of a rare disease.

"The Pakistani Brain doesn't directly alter a file or hide within storage compartments," says McAfee. "Instead, it marks sectors of the disk as bad and then, like enemy soldiers in a Trojan horse, hides itself in these 'bad' sectors. It modifies the command files on the disk, without changing file sizes or dates. This makes it almost invisible."

McAfee travels around Silicon Valley in a computer-equipped RV, a traveling viral laboratory. Like the heroes in Ghostbusters, he visits infected companies, disinfecting their computers with his special apothecary of viral cures. He has great respect for the design of the Pakistani Brain.

"The code of the Pakistani Brain is beautiful! I marvel at the elegance of its design. All of our programmers who have looked at it have had the same response: 'This is elegant! God! What great ideas! We don't admire what they did, but we do admire how they did it.' [It's] an extraordinarily good example of a virus. If you are going to do the first virus, you might as well do it right."

The brothers who created the Brain were not just smart kids, but evidently very experienced electronics technicians and programmers. Their code reflected a mastery of technique and economy of style usually gained through much practice.

Fortunately, the first virulent virus was really not that
harmful. The Brain brothers designed their virus to be sneaky, sly, virulent, and pesky, but not particularly deadly. Its primary function is simply to replicate, but it does destroy data that once lived in the sectors it marked as ‘bad.’ The Brain also has the reputation of a network crasher, causing local area computer networks to collapse because of unplanned incompatibility issues.

The virus is sneaky, because if you try to access the secret home base of the Brain directly, your computer will tell you that you are trying to access a “bad sector” and will suggest that you look somewhere else.

Basically, the Brain gets your computer to tell you, “These are not the sectors you are looking for.” And so remaining undetected, the Brain continues to execute its primary mission: mass replication.

By itself, the Brain is like an MX missile without a warhead. Recently, however, other programmers, including Dr. Cohen, have used such viruses as a delivery system, a projectile for warheads of their own design. Some versions of the Brain now carry instructions to wipe out whole data files and hard disks. Copies of these “new, improved” Brains have spread from the United States to Israel and Europe.

How do you know if you have been infected by the Brain or one of its deadly mutants? There is only one sign: a sudden change in the volume heading on your diskette. Doesn’t it seem like a good idea to remember your volume headings? That way, you will detect the change, should it occur.

THE CASE OF THE LEHIGH UNIVERSITY INFECTION

Attention to tiny details of this sort paid off for the digital detectives investigating a rash of disturbing system crashes.

The story begins at the Lehigh University Computer Center in Pennsylvania. Students checking out software from the center library were finding, much to their dis-
may, that their diskettes were not working. Nothing happened when they inserted a diskette into the computer! The diskette simply spun around and around in the drive, with nothing appearing on the computer screen.

The diskettes were not booting. It was like opening a book and finding all of the pages blank. This phenomenon occasionally happens with floppies, particularly if they have been heavily used. But in the fall of 1987, there was an epidemic of nonbooting diskettes. Somehow, something was eating the data.

It was at the loan desk that assistants handed students library floppy disks for use at home or in the lab. These student assistants were the first to notice an increase in the number of failed diskettes. They inserted one of the crashed floppies into a disk drive and attempted to read its directory (table of contents); no such luck. The floppy simply spun around and around, working about as well as a record player without a needle.

Why would perfectly good floppies suddenly stop working? The assistants had also observed an increased number of hard disk crashes in the adjacent microcomputer laboratory, a facility open to all students on campus. Were these events related in some way?

In seeking the answers to these questions, one of the students decided to examine the healthy diskettes. Bingo! The student found a major clue. One of the main files in the disk directory, COMMAND.COM, had a new date, indicating that it had recently been tampered with. Usually, the date on a .COM file changes only when a programmer has updated the file. Someone or something was changing the COMMAND.COM file on all their library disks. Very unusual and bizarre.

Curious to see exactly what was modified, the student detectives decided that the next step was to compare the modified COMMAND.COM file with an original version of the same file.

They got one of the original copies of the COMMAND.COM program that had not been circulated and compared it line by line with one whose time and date
had been modified. This was done with what is known as a debugging program.

Voila! Hiding in an empty storage compartment of the modified COMMAND.COM program was a mysterious alien program, not found in the original code.

Using special software-editing tools, the students surgically removed the "organism" from the storage compartment and studied its code carefully. They discovered some rather interesting things about it not seen in normal programming code.

The answer to the question of why the floppy diskettes were not working quickly became apparent. The virus contained instructions to wipe out the data on the floppy after it had been used four times. Nasty!

The virus did this by counting program usage via its onboard counter algorithm and then writing zeros to the first thirty-two sectors of the disk, erasing the directory and making the data irrecoverable.

The students also figured out how the virus got there in the first place. It spread when a clean PC was booted from an infected floppy and the user accessed a second, uninfected program disk.

It appears that one of the students had run an infected program on a lab machine, causing that machine's hard disk to become infected. All disks inserted into it after that were also infected. And because the floppies were shared by the students, the newly infected diskettes soon infected other machines in the lab, which infected other diskettes. The process continued until epidemic proportions had been reached.

Fortunately, the Lehigh virus never left the confines of the university. Nonetheless, it was an excellent example of a virus in action, providing the security community a detailed analysis of how these creatures worked in the world of personal computing.

THE PEACE VIRUS

Another episode in virus history demonstrates an even greater danger, the spread of infected code through
commercial software distribution.

A virus can be malignant or benign, depending on the purposes of the designer. In the early months of 1988, the Peace virus (also known as the MacMag or Brandow virus), though an example of a benign virus, was nevertheless a source of major distress to Macintosh users, ultimately affecting over 350,000 machines. The concern was not so much over what the virus did, but how it was spread. Here's what happened.

Richard R. Brandow, the twenty-four-year-old publisher of *MacMag*, a Montreal-based computer magazine, and coworker Pierre M. Zovile unleashed a benign virus to spread a short "universal message of peace" and to demonstrate the pervasiveness of software piracy. Brandow claimed the virus was supposed to "commemorate the anniversary of the introduction of the Mac SE and Mac II" by displaying a message of peace on March 2, 1988, and then wipe itself out.

To create the Peace virus, Brandow had hired Drew Davidson, a twenty-three-year-old programmer from Tucson. Personally, Davidson was against malicious viruses, calling the people who created them "attention seekers." Yet he admits that one of the reasons he created the Peace virus was to show off his skill as a programmer. Davidson said, "I didn't realize the virus would have the kind of impact it had. I just thought we'd release it, and it would be kind of neat."

The virus was first noted internationally when it was found attached to a HyperCard stack on CompuServe's Mac Forum (an on-line information service). For those familiar with Apple's Hypercard environment, it was in the form of an XCMD, and it installed an INIT ID= 6 with a name of RR.

The Peace virus also spread voraciously via computer bulletin boards and by the amazingly prolific disk swapping rampant among Macintosh enthusiasts.

The real surprise came when Brandow and Zovile's virus appeared inside a commercial product called FreeHand, a Macintosh graphics program from Aldus Corpo-
ration of Seattle. For three days, Aldus was inadvertently packaging the Peace virus and shipping copies around the globe! Once the virus was discovered, Aldus pulled back the tainted disks, but not before some got to customers.

The idea that a virus could infect a commercially distributed program sent shivers up the spines of people everywhere. James J. Mitchell, business editor for the San Jose Mercury News wrote on March 20, 1988, “Because of their fear of hidden, destructive software, many computer users have not used programs distributed over electronic bulletin boards, which are accessible through phone lines. Instead, they have bought far more expensive software, made by well-known companies and sold through retailers, because they thought it was safe. But FreeHand, a program made by the creator of some of the best desktop publishing software, clearly wasn't free of viruses, and now that hackers have seen that major software publishers are vulnerable, they may try nastier tricks.”

In this case, the nasty trick was an “accident.” Backtracking its evolution, investigators found that the Peace virus had been inadvertently passed to Aldus by Marc Canter, president of MacroMind Inc. of Chicago, a computer company that makes tutorial disks for Aldus.

When Canter was on a trip to Montreal, Quebec, he was given a copy of a program called Mr. Potato Head, a computerized version of the toy spud. Little did he know that the very diskette he carried back with him from Canada would go down in history as containing the first commercial virus, a message of universal peace!

Once home in Chicago, Canter went back to work writing training programs for Aldus Publishing on—you guessed it—a Macintosh containing the Mr. Potato Head program! The tainted program infected his computer, which in turn infected a Mac at Aldus. From here, the trail is less clear, but it went something like this:

First, the infected Aldus Mac spread the virus when used by someone in marketing. Next, an engineer used
the marketing system and infected his engineering Mac. Then, production used the engineering computer for checking the master production disk, and without anyone realizing what was going on, infected disks were soon being sold to consumers. When the consumers used the disks, their computers became infected. And those who shared their dirty disks spread the virus even further afield.

Canter believed that some of his other clients may also have become infected, clients like Microsoft, Lotus Development, Apple Computer, and Ashton-Tate.

The virus is thought to be harmless now, since it was designed to pop up on Mac screens everywhere with its universal peace message on March 2, 1988, and indeed it did! On that day, thousands of Macintosh owners were greeted by a drawing of planet Earth and a "universal message of peace" signed by Richard Brandow. It then deleted itself from existence, disappearing from screens around the world in an instant, without a trace.

Still, the stage was set for potential widespread disaster. The chances of such a phenomenon happening again were high. And next time the virus would not be so benign.

The social patterns of sharing computer systems are well established and will not change overnight. There will always be instances where someone from engineering or production will use the company's marketing computer system because it has the new 4,000-dots-per-inch laser printer, and if the company is on a local area network, infection could occur on all machines in the company in a few minutes!

For many, the Peace virus was the first and sweetest taste of things to come. It became a model for numerous other, increasingly distasteful, computer viruses.

**THE nVIR nFESTATION**

Almost immediately after the Peace virus deleted itself, a number of copycat viruses made their appearance.
Imagine turning on your computer in the morning, and instead of the familiar sights and sounds of boot-up, the machine speaks to you in a strange semihuman voice saying, "Don't Panic!" and then decides to take the day off and shuts down the hard disk for a while. A real bonus in your day.

In spite of its rather disturbing manners, the nVIR infestation never amounted to an epidemic. It was just a small-scale panic, limited to a few thousand machines. The nVIR's biggest claim to fame, like that of the Peace virus, was that it served to wake up the Mac community to the threat of viruses in general.

"What we are seeing is the natural spate of copycat-ism that I was afraid would follow the Brandow Peace virus," noted Don Brown of CE Software, a Des Moines, Iowa-based developer of Vaccine, a program designed to protect users against some viruses.

Tracking backward once again, we find the nVIR copycat viruses were based on source code uploaded into the Apple Developers Forum on the CompuServe network in late December 1987 by West German programmer Matthias Uhrlich. Uhrlich claims his code was originally intended to be educational.

More than a few programmers jumped at the chance to be virally educated. And like any class of students, a few simply copied the teacher's plans and then tried them to see if they actually worked. Others took it a step further by adding their own code. "Don't Panic!" is an example of this kind of thinking. The more malicious students skipped humor and designed their viral warheads for data destruction and other forms of computer mayhem. All viruses based on Uhrlich's code became known as nVIR strains.

To balance things out, Uhrlich later uploaded a program on CompuServe called KillVir, an antiviral program designed to remove viruses derived from this new strain.

All this fun with the Peace virus and nVIR strains was only a warm-up for the biggest hitter so far in the Macintosh community, the Scores virus.
THE SCOURGE OF SCORES

To date, the most devastating virus to hit the Macintosh world is Scores, an infection that got seriously out of control. It was originally designed for a special, malicious purpose: to attack custom application programs designed by specific individuals at Electronic Data Systems in Dallas, Texas.

Like a heat-seeking missile, the Scores virus seeks to destroy only Macintosh applications with the programming signatures ERIC and VULT. These signatures, which can only be read with programming tools, let the Macintosh match data files with applications and help perform other operating system functions. Programmers commonly use unique signatures to avoid conflicts with other software.

As highly targeted as it was, Scores quickly escaped the confines of EDS and infected thousands of Macintoshes around the world, causing incidental printing and system problems and hard disk crashes. None of these problems were planned or intentional; they occurred as a result of the virus's presence in the user's personal computer system.

The Scores virus got the Mac community up in arms. Hundreds of user groups around the world banded together to stamp out the scourge of Scores. Almost overnight, diagnostic data about the virus were spread through international computer networks. The locations and nature of the latest outbreaks were reported immediately to all interested Mac users.

The Mac community discovered that the ERIC and VULT signatures were used by EDS for some classified U.S. government applications. They found that a number of government installations, most notably NASA and the National Weather Bureau, were among the hardest hit.

The virus was written to make the targeted program dysfunctional. It lay dormant for two days after infection. At two, four, and seven days, various parts woke up and began their mischief. Two days after the initial infection, the virus began to spread to other applications. After
four days, the second part of the virus began to watch for
VULT and ERIC applications. Whenever VULT or ERIC was
run, the virus caused the system to bomb after twenty-
five minutes' use. After seven days, the third part of the
virus kicked in. When the targeted application was run,
the virus waited fifteen minutes, then caused any at­
tempt at writing a disk file to fail. If no writes were
attempted, the application would bomb anyway after ten
minutes.

Over half a dozen cures for Scores were developed
within a month. Some were so sophisticated that they
automatically repaired programs damaged by the virus.
A few programmers discovered that they liked being
computer doctors, and whole new careers were started.
Most members of the community, however, simply
wanted to rid themselves of the disease.

**THE IBM ELECTRONIC MAIL COLLAPSE**

People in the personal computer community were not
the only ones crying in their keyboards over viral attacks.
In December 1987, an unexpected Christmas present
was delivered to the electronic mailboxes of all em­
ployees of International Business Machines Corporation.

Written as a prank by a West German student, this
virus began in a European academic computer network
(Bitnet) and jumped through the electronic gateways to
five continents and into the internal electronic mail
system of IBM.

In the IBM system, a holiday message promised to
draw a Christmas tree on the screen if the user would
type the word *Christmas* on the computer. When anyone
did, it drew a tree but it also sent a copy of itself to all of
the other network mail addresses kept in the user's
electronic Rolodex file. Along with a very primitive tree
(made of capital X's), a message was displayed:
Once opened, the program rarely accepted commands to stop. Anyone who turned off the terminal to try to stop the Christmas message lost electronic mail or unfinished reports not saved in the computer. It infected so many machines that it brought IBM’s global electronic mail network to a halt, disrupting the system for seventy-two hours. To purge the message, plant officials were forced to turn off internal links between computer terminals and mainframe systems.

A counteractive virus was written to follow and destroy the Christmas Card virus and then self-destruct in mid-January. The electronic mail Trojan horse was virtually eradicated by December 14, 1987. Its creator was tracked down and barred from access to his system.

**THE APPLE E-MAIL STORY**

In December 1987, Apple Computer found a virus in its electronic voice mail system that shut down the server. Every time Apple restarted the system, the virus resurfaced and shut it down again.

The server is like a mail carrier—it delivers the right messages to the right people. The virus was what Apple called a sneak virus, hiding its tracks while searching for its prey. It lurked in the memory, waiting for the right moment, then whammo-bammo, it would shut down the server again and again. The virus succeeded in erasing all of Apple’s electronic voice mail.
While the server was knocked out, the virus searched specifically for Diskfit backup files. Like the reserve troops in an army, these files are brought in when the primary files have been seriously damaged or destroyed. As the virus found these backup files, it erased them, leaving nothing but the file's icon.

Apple was concerned that the virus would get it into the company's massive electronic mail system as well. Apple had hundreds of Macs, all interconnected by E-mail.

After Apple reformatted its sixty-megabyte voice mail hard disk, it disconnected all the hard disks on the sales floor from the electronic mail system. The company saw the public sales area as an open wound, ready to be infected.

The reformatting of the hard disk seemed to do the trick. But it was enough of an inconvenience for Apple to start taking viruses seriously.

**THE ULTIMATE VIRUS ARTICLE**

Future viruses may not kill our computers. They may kill us. At least, a careful look at the history of hackers may lead you to that chilling conclusion.

Every technology has its hackers. A technology only has to become accessible to a sufficiently large number of people for hackers to appear. The hacking probably starts as soon as the science begins to be taught in universities, maybe as soon as it is discovered in the research community. I suppose it all depends on how the term is defined. Hobbyists and commercially unsuccessful small enterprises might be included or not.

The term probably should include any effort that employs a new technology in a socially disruptive way. That's a negative connotation that has become strongly associated with hackers. Although that disruption is representative of only a small number of individuals, it is what generates all the attention for them after all.

Hams have hacked radio hardware, and CB users hacked out a popular culture. Telephone hackers hacked
out and touched Ma Bell—for free long-distance calls. Television hackers have interrupted satellite transmissions and created community-access television programs. There have been rocketry hackers—at least one of whom has seriously contemplated manned space flight. Early chemical hackers discovered beer and other forms of anesthesia, and later ones created designer drugs. There have already been nuclear hackers. Students have designed nuclear weapons for science projects that have prompted investigations by national security agents. Hackers have written pamphlets telling terrorists how to do it.

Now computer hackers have come up with viruses, worms, and other nuisances that are stirring up the data-processing industry. It's only another version of the phone phreaks, of overpowered CBs, of out-of-tune broadcast antennas, of outlaw radio and TV stations. The basic phenomenon is constant. It has to happen because there are far more people who want to experiment with science than can be trained, funded, and monitored. It's as much human nature to have that curiosity as it is for a certain number of the experimenters to find asocial outlets for their knowledge. Some of the same people who have the hacker's strong curiosity also see the successful perpetration of an asocial deed as a challenge, as an achievement.

Where does the danger enter into it? Well, the next generation of hackers will probably be genetic experimenters. Like their chemical-hacker counterparts, they may begin by seeking some form of counterculture amusement. And that's where the danger lies. From seeking amusements, it only takes one bitter moment to turn a failed experiment loose into the world at large.

A month or so before COMDEX [Computer Dealer Exposition], I was discussing computer security with Buck BloomBecker of the National Center for Computer Crime Data. The viruses we had seen up to that point were relatively limited, but we had read speculation of killer viruses that would damage hardware in the near future. I suggested to Buck that the viruses would have far more devastating effect if they were distributed more efficiently over large networks through electronic mail sys-
tems. It was just a logical extrapolation, given the available technology and assuming a desire to cause the maximum disruption.

Apparently, others were contemplating the same possibilities. This past month has seen that type of virus distributed over a large national network: Internet. The biological designer virus is the same type of extrapolation. I'm not suggesting that it may happen. I'm saying that it will—and maybe already has. One thing is for certain: when it does happen, there will be cover-ups that make computer security cover-ups pale by comparison.

We would probably already have amateur nuclear weapons developed by hackers if it weren't for the difficulty of obtaining fissionable material. Biological materials, required for genetic engineering, don't demand the same enormous capital investment required to refine plutonium. So take my advice: be happy, don't worry—it's going to happen anyway. Just be careful what you download.

—Wayne Yacco

JOHN McAFEE AND HIS TRAVELING VIRAL LABORATORY

To curb the spread of these new and ever more virulent invaders, a new breed of computer professional was born—the viral detective. The first of these digital sleuths was John McAfee. McAfee is a virus expert and creator of antiviral software packages C-4 and Tracer, founder of the first antiviral trade association, and president of Interpath Corporation in Sunnyvale, California.

From outside, the twenty-seven-foot motor home known as the Bugbuster looked like a traditional vacation RV, complete with "Good Sam" and "Retired & Loving It" bumper stickers and scenic state decals crowding the windows. The inside of the vehicle, however, presented a different and highly unusual scene, a traveling viral laboratory equipped with half a dozen microcomputers and several hard disks.
John McAfee takes the Bugbuster on house calls to San Francisco Bay area companies and colleges, dispensing digital vaccines and collecting a rogues' gallery of viral specimens along the way.

I accompanied John to the College of Alameda, near the Navy's collection of nuclear vessels, where the student computer lab was infected. John's job was to eradicate the bugs.

Before we got under way, John demonstrated the impressive virulence of the Pakistani Brain virus. We watched it replicate before our eyes on one of the Bugbuster's traveling computers. It was amazing to see a virus in action.

The journey to Alameda was a nonstop question-and-answer period on computer viruses, where I had a chance to interview McAfee about the viral infections he has encountered. He answered some fundamental questions about viruses and how they gain control of computer systems.

**AL:** What are the computer processes that viruses access, allowing them to take control of the machine?

**JOHN:** The operating system in personal computers provides the means to access what are called low-level processes. These are processes that allow you to do things like format the hard disk or read the boot sector, processes that most computer programs never use and couldn't care less about. Such functions are usually reserved for utility (maintenance) programs.

**AL:** How do viruses access these low-level processes?

**JOHN:** Viral codes often contain operating system instructions for accessing low-level processes; it's what makes viruses far different from other programs. It is through the use of these processes that a virus can erase your hard disk, for example.

**AL:** How do you use this knowledge for detecting a virus?
JOHN: It is possible to detect a hidden and buried virus by designing a routine that looks for these low-level operating system instructions hidden within the code of your executable programs (like using a word search function in a word processor). Because the virus is embedded in your program, its code is listed as part of your program code.

AL: What are these low-level operating system instructions?

JOHN: They are called call interrupts. After an interrupt is issued (usually by a utility program), it branches to a low-level routine that performs one specific function. This low-level routine has a location address in memory called an interrupt address vector. There are many different types of interrupts, and they are all accessed via their unique address vectors.

AL: Can you give us a specific example of how you apply this knowledge in your own antiviral software?

JOHN: A program will issue, say "Interrupt 13." After Interrupt 13 is issued, it goes to the Interrupt 13 vector address, a specific place in memory where that interrupt code resides. Our software [C-4 and Tracer] goes in and changes those interrupt vector addresses, so that when a program issues an Interrupt 13, instead of branching to the operating system, it branches to us. The program doesn't know that we are doing that. In this way, we trap all those addresses so that they are now within our code. As in the case of tracing a phone line, when Interrupt 13 is accessed by the system, we look at it and say, "OK, you are accessing Interrupt 13; who are you, and what the hell are you doing?"

AL: So what happens if we install your antiviral program and run a program that has valid reasons for using interrupts, like a utility routine?

JOHN: If the program accessing the interrupt vector looks OK to us, we save the original vector interrupt address, we branch to the real interrupt vector ad-
dress, and the operating system goes about its business.

**AL:** What happens when you catch a virus accessing an interrupt vector?

**JOHN:** If an interrupt vector access looks suspicious, we say, "Wait a minute, you are trying to write something to the boot sector! That's not a kosher thing to do! We are not going to let you do that." We then display a window to the computer user, saying, "Someone is trying to do a low-level access to your boot sector." The virus, you see, thinks it is talking to the operating system, but we have intercepted it, and it is actually talking to us.

**AL:** Well, it does take a sneaky approach to catch a sneak, eh?

As you can see, we arrived in Alameda with a better understanding of viral workings. The people at the college were glad to see us. It turned out that their computer lab was infected with a version of the Pakistani Brain virus. John quickly dispatched it with his bag of antiviral tricks, but not before collecting a copy of it for the traveling lab. Intrigued by the process, the college lab instructor decided to return with us to the RV.

By coincidence, it turned out that he had been trained in computers by an old college buddy of mine, Fred Cisin. A maverick investigator of interesting phenomena, Fred studied the earth's Van Allen radiation belt at Goddard Space Flight Center. Fred taught me how to weld plastic using radio waves when he created the Blueberry Muffin, the first perfectly round water bed. But that is another story.

With the viral mission complete, it was time for a break. I asked the crew if they would like to meet Captain Crunch, alias John Draper, King of the Nerds, an Alameda resident and one of the original phone and computer hackers. I thought McAfee would like to meet someone who knew a few viral designers. Crunch was
famous for his adventures with blue boxing the phone system, and later played a prominent role in the evolution of the original Apple Computer through his friendship with Steve Wozniak, Apple's co-founder and prime nerd genius. Besides writing the first word-processing program for the Apple II (Easy Writer), he had designed the Telephone Board, a card for the Apple that gave it awesome computer- and phone-hacking capabilities.

One of the things that got Crunch in deep trouble was when he hacked the White House security system. He woke up Richard Nixon in bed and told him there was a shortage crisis in California. When Nixon mumbled, "What kind of shortage?" Crunch replied, "We've run out of toilet paper!" Typical nerd humor, not always appreciated by those in power.

John was very interested in meeting the Captain, and the Captain wanted to see the viral laboratory. The Captain met with us in about twenty minutes. Surprisingly enough, he knew very little about viruses. He was, however, interested in hearing what John had to say and watched the Pakistani Brain demo intently.

After the demo, Crunch invited everyone to the inner sanctum of his hacker haven for a show-and-tell session. There he showed us some amazing color graphics and video images on his maxed-out Mac II system. Personal desktop video, the synthesis of computer and video technologies, is definitely the wave of the future.

The trip back to San Jose was relaxed and jam-packed with more viral data. Here are some of the more interesting tidbits I collected that afternoon from John McAfee:

**AL:** How was the Pakistani Brain virus discovered?

**JOHN:** People noticed it when funny things started happening within their computers. They would install a new version of their operating system, for example, and yet the old version still seemed to be running. All the new commands didn’t work. Typically a computer consultant would run into this situation and exclaim, "My God! I’ve never seen anything like this."
Ultimately, after screwing around with it enough, he would finally do the right sequence of things and see the boot sector, where it reads, "Pakistani Brain." "What the hell is this?" he would say. That is how the Brain is usually discovered.

AL: What are the newest viruses you have investigated?

JOHN: We got a call from a lady last week who was almost in tears. For months she had been running lots of programs, and they would just disappear. She thought she was going crazy.

"I kept on making sure I was not deleting these programs," she said. "I know that I am not that experienced, I've only been using computers for a couple of years, and I had no idea what was going on."

It turns out she had .COM infector. This type of virus infects any .COM file, and when run, the .COM file disappears, but only after a certain number of executions. A lot of people initially think they have done something wrong.

I got a call from a guy this morning who has a brand new virus. I call it the Laughing virus. It's a really slick virus and apparently a boot infector. When he runs a program, the screen goes blank, and then on-screen cues appear. They would then disappear, and the screen would come back. Sometimes he would have a line of garbage characters at the bottom of the screen.

Suspecting that this strange behavior might be due to bugs in some new software, he removed the program from the system. The problems continued to occur. And then a new symptom appeared. Every time he booted his system, he heard what sounded like music. "Really weird music," he said. "It sounded like somebody laughing."

His system had been working fine until he loaded a public domain program (which he had bought). Apparently his system became infected by this shareware program.
A: A couple of doctors at the AIDS conference in Stockholm said that AIDS has antecedents from thirty or forty years ago. Is the same thing true with computer viruses?

J: I can almost guarantee that there were no such things thirty or forty years ago, because viruses are unique to the personal computer world. They make no sense in any other environment. A virus on a mainframe is nonsensical. General Motors, for instance, doesn’t go around passing disks to Ford. You have a mainframe environment that is all in a room somewhere. If you are going to infect it, just plant a time bomb. Why bother designing a program that moves from computer to computer when there is no chance of that happening?

A: Can viruses infect data files as well as programs?

J: The virus can only move from computer to computer on an executable piece of code. Dr. Cohen talks of viruses and data. Now, that is all good and well, and theoretically you could put a virus in a piece of data, and theoretically it could get control of the program. But in the real world, you will never see it happen. I think the chance of that happening is about as likely as getting hit by a meteor. It’s possible, but it’s not something I would worry about.

How would a virus in data ever get control? Even to migrate from the data, it has to get control. And for it to get control, you have to execute it. That is the problem. You never execute a data file. First of all, the operating system won’t let you. You can’t type “LETTER.TXT.” The operating system will come back with “BAD COMMAND.” To execute, it has to be a .COM, .EXE, .SYS, or .BIN file. It just is not reasonable: No one will write a virus in data, because the virus might get executed once very fifteen thousand years or something.

A: What do you think a more likely scenario might be?

J: Viruses that attach to programs. Programs move from computer to computer because people move
their disks from computer to computer.

The environment of the personal computer world is very conducive to the spread of viruses. There is a lot of transfer of diskettes. I use your computer because you have a laser printer and I don't. So every time I want to print out a file, I use your system with my disk. If my disk is infected, your system becomes infected. If your system is infected, everyone who uses your laser printer goes away with an infected disk. Then people put that disk back in their system. Then their system is infected. That is how it happens.

AL: Weren't there a few viruses designed in academic circles?

JOHN: There may have been some students who created one somewhere for a term paper, but it was never very effective in terms of transmission, and it never got anywhere.

AL: Haven't viruses been around for a long time?

JOHN: Prior to 1986 people used the terms viruses, bugs, worms, and Trojans synonymously. Trojans have been around for years and years. They are little pieces of program that go in and destroy your software. They might even include a time bomb. But they are not capable of living, that is, finding a host that they can attach themselves to without damaging the host and continue the cycle of life. That is what I call a virus today—totally different from the time bombs that people would design to go off in four or five years.

AL: What do you consider to be the biggest problem caused by viruses?

JOHN: One of the biggest problems with viruses is that you can be infected and not know it. Because you don't see it, it isn't doing anything to you, it hasn't activated yet. And many viruses don't activate for years, so you've got a virus, you are infecting everybody you are coming in contact with, but everything seems fine, nothing is going wrong, your WordPer-
fect program runs perfectly. Then a year down the pike, you turn on your system, and everything is garbled or, even worse, you have a numeric virus that has been subtly modifying your numeric data for the past year and a half, and you haven’t noticed it. And all of your backups are screwed up as well.

Most people who have been hit by viruses only find out about it after a lot of data have been destroyed. They don’t know they’ve been infected, so they don’t think there’s a problem until it is too late.

AL: Do you think someone was inspired by the AIDS virus in designing a computer virus?

JOHN: It certainly is possible.

AL: How useful do you find biological analogies? Could reading about an AIDS virus help you to design a computer virus?

JOHN: Epidemiologically, the biological and computer worlds are identical in that the transfer mechanism and the transfer statistics are valid. They hold. There could be a mapping of biological growth. But the mechanisms of reproduction are not identical.

When it comes to biological viruses, we are talking about a very complex method of reproduction. When it comes to computer viruses, we are talking about a much more orderly and constrained method of reproduction.

But yes, there are a number of parallels. First, the virus must replicate to live, to carry on the species. The most successful virus will replicate the most.

In order to replicate, there must be an acceptable host. In other words, a receptive host. Not all programs are receptive.

Finally, in order to replicate, it must get out of the machine that it is in. It has got to be capable of identifying the environment that it wakes up in. Every time an infected program executes, the virus wakes up: “OK! Where am I? What’s my status?”

The first thing it does is ask, “What’s my environment? Hmm. I’ve got two floppy disks, a twenty-
megabyte hard disk, a Bernoulli box, 640K memory. All right! I can live in here!

So then it says, "Are there any hosts in here?"

And then it goes to all the floppies and hard disks and says, "Is there an acceptable host?" And almost always, there will be one. Usually about one in ten programs will be acceptable. Then it says, "Great! I've got an acceptable host!" And then its internal logic says, "How many acceptable hosts can I safely infect today?"

If you have 500 programs that are acceptable hosts, it won't infect them all immediately because you, the operator, will notice and say, "What is going on here? I just ran this program and nothing is happening!" So generally, the virus infects something in less than a second—so that you, the operator, will not notice anything unusual. So it will infect as many hosts as it can in less than a second. And then it will say, "I've done my job. Is it time to activate?" In other words, "Is it Friday the thirteenth yet? Have I infected a hundred times already? Is it 1Jan90"—or whatever. If so, it passes control over to its host application program. The application program doesn't know anything has gone wrong. It just thinks it has been called by the operating system and goes about its merry business. That's how the virus lives. That sequence is constant, irrespective of the type of virus.

Now, within each of those blocks, what it does will vary tremendously. Nevertheless, that sequence will remain constant.

This is ideal, of course. Some viruses are so dumb they keep reinfecting the same program over and over again, like the Jerusalem virus. There is no rule of thumb.

If you are a virus and it is your day to activate and destroy, the only thing you would want to do is look for a floppy in one of the drives to infect. One final last fling before you go out, because when you acti-
vate, you are going to destroy everything you've infected, anyway. That's the final activation phase. "This machine in which I have been living for the last few years has been a good home, and now it is going bye-bye! And I am going bye-bye with it!" It then activates, and everything goes away.

Now, not all of them will self-destruct. Some are so slick that they will destroy almost everything but stay alive themselves somewhere, say in the boot sector. "You guys are all going, but I am staying!" But again, that is not a rule of thumb either. Keep in mind, we are not dealing with a physical law here, and this is where I differ radically with Dr. Cohen. We are not dealing with the law of thermodynamics. We are dealing with the human mind. And the law is whatever the hacker decides the law will be. You can't say this is the law of viruses. The law is whatever he decides to design. And that is what you have to respond to.

If a guy decides to create a virus that replicates ten times and then destroys itself, then that is what you have got to deal with. You can't deal with some theoretical law. You deal with what is out there destroying data.

AL: So you are dealing with something as variable as the human mind?

JOHN: That is correct. That's it.

AL: How would you describe the typical viral designer?

JOHN: There is no typical designer. They range all the way from hackers who have thrown something together overnight to geniuses who can design absolute masterpieces like the Pakistani Brain. Some of them are not even aware of all the DOS facilities they could be using. We've seen viruses where we've said, "God! Who programmed this? I'd never hire this character. It took him four hundred instructions when he could have done it in three!"

AL: Do you see us developing a general defense against viruses?
JOHN: Yes. In five or six years, there will be no need for specific antiviral products. The environment will force computer vendors and operating system manufacturers to place in the operating system code, and perhaps in the operating system itself, the features that are necessary to prevent viral replication. That would be an easy thing to do, so I think that will happen.

AL: Like developing an immune system for computers?

JOHN: That's right. Are you familiar with the UNIX world? In the UNIX world, especially in government applications, there are things called Trusted Computing Bases (TCBs). There are different levels of these things. They are imbedded within the UNIX operating system. Now it used to be that these were special applications that were jerry-rigged together to throw on top of everything to try to provide a degree of security. It was not the way to go. So now, there are TCB operating systems. If you are doing a bid for the government and they want to buy a computer for a trusted defense environment, you bid TCB UNIX. And that is the sensible way to do things.

If you have an operating system you are trying to protect, put the protection in the operating system. If you have a computer you are trying to protect, put the protection part and parcel into the computer. I see that happening. That is the way to go. But in the interim, in the next four or five years, people like myself are going to be extremely busy, and the types of products we are developing will have a fairly widespread use.

AL: What is the scope of viruses nationwide? How many corporations, how many people have been hit?

JOHN: I couldn't even hazard a guess. We've been contacted by over 200 companies that have been hit. And we've talked to a lot more people who want the program and ask specific questions and may or may not be infected. But if we have heard of 200, and we are a dinky little outfit of programmers working out
of our homes, there have to be an awful lot of infections out there.

AL: Have you spoken to others like yourself?

JOHN: I’ve talked to all of them, actually—all of our competitors in the antiviral field. Quite frankly, most of them have not collected much information on viruses. We have a bulletin board system, and for a while we were offering rewards to people for turning in viruses. That is a different thing. We don’t count them as infections. People turn in something on the bulletin board—God knows where they got it, we couldn’t care less—but that is how we initially started collecting viral information. We got a jump on the world in this way.

But quite frankly, not very many people out there have any information at all. Most people are flying blind. I’d say that 75 percent of the companies that are writing antiviral software are writing it in a vacuum. They are writing it on fairly sound principles. They are good software engineers, and they think, “Well, gosh, how would a virus act? It must do something like this.” And they write a program to counter that. And that’s OK. But they simply don’t have the actual viruses.

We’ve had quite a few sessions ourselves looking at actual viruses. Again, they do not follow natural laws. How we think things might happen is not necessarily the way the world works.

AL: What institutions are most vulnerable to viruses?

JOHN: If I had to generalize, I would say the institutions with the most personal computers. We are talking statistics, nothing else. If you have only one personal computer, you could handle the problem. But if you have fifteen thousand personal computers, you probably have fifteen thousand users. And that means that the probability of one of those users having a computer at home which he or she uses to access a bulletin board, the probability of that person coming in contact with an infected piece of
software is pretty high. It is pure numbers; that is all it is. If you have two people, it is twice as likely as one. If you have fifteen thousand, the probability is pretty high. It's just a fact of life.

Remember that the organization most susceptible probably also has the highest degree of security. If the army knows that personal computers are inherently not secure, even without viruses, it is unlikely that any personal computer is going to be connected directly to the master control satellite link in Sunnyvale. That is unlikely. So it is unlikely that our national defense will suddenly, at least globally, disappear. However, as you decrease the security requirements and go down to, say, the Treasury, it is a lot more lax. I would expect the IRS to have problems.

Then, when you get down to our financial institutions, which are pretty much the backbone of America, they have little or no security at all on the personal computer side. And probably much of their transactions go through personal computers, so there I would say the risk of a serious viral infection is very high.

**AL:** Don't they have backup tapes?

**JOHN:** Yes, but keep in mind that viruses can be dormant in your system for years. All your backups could be infected. Using one would put the virus back in your system. Backup strategy needs to be rethought in the age of viruses.

The bottom line and most vulnerable is American industry. That is where the security is least secure.
Tales From the Cyberpresent

Carrie was excited about her new purchase. A graphic artist for years, she was finally able to afford her own personal graphics computer. Now she could work at home designing logos, letterheads, and advertising images for her clients.

One day she was racing to complete a difficult logo design. She had spent days laboring over the image, getting just the right look for her client.

All of a sudden, her treasured logo disappeared from the screen and was replaced by the following message:

Something wonderful has happened.
Your Amiga is alive!!!
and even better
Some of your disks are infected by a VIRUS
Another masterpiece of the Mega-Mighty SCA

Carrie was scared and felt personally violated, as if there were little viral creatures crawling all over her
disks. Frantically trying to figure out what to do, she thought, "Will they disintegrate or completely burn out my computer?! Will I lose my client? Oh, God! What did I do to deserve this? What will I do?"

Carrie had just been attacked by a virus. And it felt very real.

Fortunately for Carrie, the Amiga virus delivers its message and then disappears, never to be heard from again. But she didn’t know that.

Most people who are victims of viral attacks don’t know what to expect and usually fear the worst. They have no idea what has happened or what the virus is doing and no way to evaluate it. Their personal computer has been invaded quite literally. Something alien has gotten into it and taken away their control.

It truly is an attack and is usually taken quite personally. In Carrie’s case, she was afraid all of her work had been destroyed and was careful not to touch the machine for fear of getting infected herself. She suffered fairly severe depression and was paranoid about her computer’s security until a friend gave her a vaccine for the virus and some comforting words.

Approximately 150,000 people have had an experience like Carrie’s in the last year. They have been dramatically educated in the importance of personal computer security. Unfortunately, most of us don’t learn until we have had an accident.

There is an old saying, "If it ain’t broke, don’t fix it." Let’s say you haven’t had any problems so far with your computer, and you have suddenly won a hundred dollars in the lottery.

I ask you, "Do you want to go out for an outrageous first-class dinner, or do you want to buy a security product for your personal computer?"

You might ask me, "Well, gee, is this security product going to make my computer do anything it isn’t doing now?"
"No," I answer.
"Will it speed things up?" you ask. "Is there any benefit?"
"No, it's going to slow things down," I respond. "The only benefit is that your computer might not break as often."
"Will it certainly not break as often?" you query.
"Noooo . . . it might not break as often," I answer.
"Well, let's go to dinner!"

And then the next day you lose all your data from a sneak virus that had been secretly living in your system for the last three months. Suddenly you feel raped. "My God! I've lost ten years of work," you cry. This time you look at me and say, "I'll skip dinner for a week and buy three layers of security."

"I don't blame anybody for not taking this as seriously as I do," says Dr. Cohen. "It's a matter of education. It's also a matter for the people who make computers and sell the operating systems and software. When you buy a program from a manufacturer, it should protect itself."

In the meantime, it definitely pays to invest in system protection yourself. Feeling victimized is no fun—no fun at all.

HANDLING CHAOS

When your computer system, be it large or small, goes down, chaos reigns. Depression pours into your experience and settles into your mind like wet cement. Life takes on a more somber tone. All those great ideas and thoughts that you managed to capture in the moment while sitting in front of your computer—gone! Unless you can somehow recreate them.

I know. I've just lost the last three hours of writing for this chapter. The best three hours of writing in months. Gone! And no clear reason why. I felt I had done everything right, but it's still gone, so I'll surrender. I had to surrender all those changes I made, the technical trans-
lations, illustrations, and so forth. Surrender even the hope that I could remember it all again. At least I have this paragraph of prose, reflecting my mood as I recover from a hard disk crash.

When you’ve just lost your precious book, movie script, play, poem, or formula, your attention is usually focused on getting your material back on the screen. You ask yourself questions like “Are there backup files? How can I see them? What’s the procedure?”

In the haste to restore your files as quickly as possible, you can easily overlook or destroy many clues as to what went wrong. This is comparable to police arriving at a murder scene to find investigators all over the room, handling the weapon, stepping on blood splats and other evidence, before fingerprints, biosamples, and imaging data are collected.

Many in the microcomputer world feel strongly that viral hackers ought to be severely reprimanded for the pain they cause to others. Bill Atkinson, one of Apple’s premier programmers and creator of Apple’s Hypercard, was attributed to saying this of programmers who unleash viruses:

They should be roasted over a spit for about sixty hours, while their skin is ripped off with pliers, their family taxed out of existence for sixty years. And then they should be taken out and shot.
"The government is scared and is going to hang a few people. They want somebody to put on the cross. It gets real serious when nuclear warfare software is threatened. It is not a good time to be a virus maker."

—Steve Dompier,
Chairman of the Board,
Island Graphics Corporation

Most virus makers are not interested in starting a thermonuclear war. Their motivations are usually much more mundane, like warped fun or revenge. Nonetheless, the law enforcement authorities in this country have taken the viral threat very seriously, going for convictions when possible.

THE FIRST SHOT IN LEGAL DEFENSE

Take the case of Donald Gene Burleson, a forty-year-old programmer. On September 19, 1988, a Fort Worth jury deliberated for six hours before finding him guilty of
planting a virus in the computer system used to store records at the USPA and IRA Company in the Dallas/Fort Worth area.

As the insurance and brokerage firm's director of computer security, Burleson was well positioned to wreak havoc. On September 19, 1985, he was fired for alleged personality conflicts with other employees. Two days later, when 168,000 payroll records disappeared, Burleson was suspected immediately.

"The virus," said Tarrant County Assistant District Attorney Davis McCown, "was activated on September 21, 1985, by a time-delay trigger. There was a series of programs built into the company's mainframe computer system as early as Labor Day [1985]. Once Burleson was fired, those programs went off. The virus held up company paychecks for more than a month and could have caused hundreds of thousands of dollars in damage to the system had it continued."

From the DA's perspective, the most convincing evidence against Burleson was an oral statement made by one of his ex-buddies at the company. His friend, working overtime trying to fix the damage caused by the virus, gave Burleson a call and said something like, "You're mad at the company because you've been having all these problems and they fired you. Now I have to work overtime to fix what you did. You are out there playing golf, and I'm working in front of a terminal Saturdays and Sundays!" Burleson felt some sympathy for his overworked buddy and told him the secret name of the virus file. He said that if that filename were taken out, the virus would be killed.

The evidence the jury considered most significant was information showing that the construction of the plague programs could be linked to Burleson's terminal, under his account, on a day when other employees saw him at work. He had a private office with a private terminal and apparently his nefarious activities were going on all day long.
It wasn't as if somebody had slipped into his office while he wasn't there and had quickly compiled a program to destroy company payroll records. The work on the virus was interleaved with his regular work. The work log of the day clearly showed how he was working on the virus, then doing legitimate work, then the virus, and so on, all day.

The jury found this more impressive than the oral statement. While Burleson's ex-buddy could have lied, the computer records were tangible evidence showing what had actually occurred.

According to McCown, Burleson didn't have a defense at first: "He was kind of wandering around through excuses, like a power failure, then he killed two birds with one stone by claiming that the guy who framed him was the guy who made the oral confession. He lied about both."

The defense lawyers showed how it would have been technically possible for someone else to use Burleson's account. His logs were saved in a text file, and text files are editable. So if the framer had created the viral programs at his own terminal, he could have saved the log to a text file and then used a word processor to change all the entries from his terminal to Burleson's terminal, making sure that time/date stamps lined up and did not conflict. A difficult but not impossible scheme.

According to Dr. Fred Cohen, however, there is no way the prosecution could prove anything with computer information. From his expert perspective, it's too easy to doctor the data.

Says Cohen, "I could write a virus that runs through a system and finds Burleson's account. It creates this thing that makes it look as if he did it, destroys all traces of itself, and then goes and spreads from his account throughout the world. In fact, the system administrator doesn't even have to do that. He can simply plant it there and allow it to spread. In fact, in that particular case, my understanding is that there was a previous, let us say, disagreement between other employees and this em-
ployee. So there may be others with motives, and certainly anybody has an opportunity to make it look as if this guy did it."

The prosecution, however, had more than one piece of evidence linking Burleson to the crime. One of the most interesting elements of the case was a huge chart assembled by the prosecution, linking all the evidence together into one cohesive story.

Says McCown, "Most of the jurors told us that, without the chart, they couldn't have followed the proceedings. It had information on it like what Burleson did on a certain terminal on a certain day, and how it linked to a series of destructive programs that were found in the computer, and how that linked to actual activity that set off the virus at three o'clock in the morning on September 21. So, anytime we were talking about something, no matter what witness, defense, or state, you could point to the chart and say, 'This is what we are talking about, right here!'"

McCown plans to use such a linking chart in future high-tech prosecutions. Even after the trial was over, he kept adding more things to it, improving his comprehension of what actually occurred.

Burleson’s motivation for destroying company payroll records apparently stems from the fact that he is a member of the tax protestor movement. He is of the opinion that taxes are unconstitutional and was actually suing his employer for withholding taxes from his paycheck. Burleson was getting into disputes over this issue with the man in charge of the company payroll.

Burleson was using the company computer to print up his legal pleas, which the company found offensive. That’s what led to his being fired. The company wanted him to drop the tax protestation and drop his suit. The situation reached critical mass when Burleson began printing up another lawsuit against the federal government on the company computer.

His anger was not so much directed at the company as it was toward the man in charge of payroll. Burleson’s
little virus made sure the payroll checks didn’t go out, bringing the wrath of the company employees onto the payroll department.

The Burleson case is unique in that it is the only computer virus case brought to trial so far. McCown claimed that his office had never indicted other cases involving the destruction of records. He said the case went as far as it did because Burleson had a bad attitude.

Burleson's claim was that the DA's office was trying to make an example out of him and get the publicity. McCown says it was just the opposite, more like “Let's work this thing out so we don't have to try it.”

In any case, it is a warning to potential viral designers. The courts are saying, “This is what will happen to you if you try to do the same thing.”

In actual practice, however, it is extremely difficult to get a conviction for such a crime. It took McCown three years to get the prosecution's case organized and two weeks to present the evidence before a jury. Burleson's case was relatively straightforward compared to today's assorted and more complex viral activity.

“In the past, prosecutors have stayed away from this kind of case because they're too hard to prove,” McCown told the press. “They have also been reluctant because the victim doesn't want to let anyone know there has been a breach of security.”

As far as future cases are concerned, the Burleson trial has opened the door to potential prosecution for viral crimes. McCown and his team have pioneered tracking techniques for investigating complex high-tech crimes and are ready to use them again. The DA's office in Fort Worth has been deluged with calls from people interested in making a case. Prior to this trial, victims of viral crimes had no legal recourse available to them. Now they have a chance, provided motives are reasonably clear.

DIFFICULTIES IN PINNING THE RAP

It is virtually impossible to prove that a particular person wrote a specific piece of computer code, espe-
cially if the viral designer does not wish to be known. To complicate matters further, modern viruses are often assembled from pieces of older viruses created by other people and then inserted into the computer networks and user groups. In a matter of days, the virus can be infecting thousands of computers, with no easy way of tracking its origin. It is a prosecutor’s nightmare.

The modern viral designer may actually assemble pieces of code written by others to create a whole new virus. It can be compared to assembling different constructions from an Erector Set. With today’s technology and what has already evolved, we could actually create a designer virus construction kit.

If such an activity were legal, the ads for a virus kit might look like this:

**Designer Virus Construction Kits**

*available only through the Weapon Shops of Isher*

**Productive Tools for Personal Response**

Why be a victim of any person or bureaucracy? With our Designer Virus Kits, we offer guaranteed protection of your individual rights. By employing the power already available in your personal computer, you too can live a truly ruthless life, free of the annoying distractions and obligations imposed by power-hungry groups and authority figures.

---

**WARNING:** Misuse of our Viral Designer Construction Kit could easily result in the loss of your liberty. Appropriate use requires a high degree of self-understanding and restraint.

---

**THE DESIGNER VIRUS CONSTRUCTION KIT:**
- Modular components for easy adaptation to your needs
- Kits available for all popular computers
- Repairs your system when not in offense or defense mode
- Requires some programming knowledge
SAMPLE SELECTION OF DESIGNER MODULES AVAILABLE NOW:

Replication Delivery Modules
These modules allow your designer virus to be spread far and wide.
- Boot and .COM infectors
- Random-access memory infectors
- Read-only memory infectors
- Laser printer RAM infectors

Message Modules
For delivering uncensored information to large groups of people. Optional self-destruct upon delivery of message.

WARNING: Users may be subject to criminal prosecution and severe penalties.

Warhead Modules
For destroying computer systems remotely. Subcomponents include:
- Multiple warhead designs for destroying many systems simultaneously
- Targeted warheads aimed at eliminating specific information within specific computer systems
- System interrupt routines for total file erasure
- Timing algorithms for coordination with other events

Telemetry Probe Modules
Find out what others have on you! Allows your virus to collect information stored in other computer systems and send it back to you without being detected. Submodules include:
- Mainframe access penetrators
- Password protect descramblers

Remote Defensive Modules
Protect your virus while it is in other computer systems. Designed to defeat many of the offensive attacks mounted against your viral probe.
Data Manipulation Modules

Are you being denied credit unfairly? Try employing one of our latest DMMs. We have special modules designed to modify your computer records in:
- The Department of Motor Vehicles
- Law enforcement agencies
- Law creation organizations
- Government taxation bureaucracies
- Consumer credit systems

Personal Power is here to stay.
And our Designer Virus Construction Kit is here for you!

Call us NOW
1-800-VIRUS 4 U

CONSTRUCTIVE VIRUS APPLICATIONS

Computer viruses are here to stay. The meek shall run and hide, the bold will embrace the new form of software and see how it might advance their quality of life as well as that of their fellow humans. It is important that computer viruses not simply be used by the Lex Luthors and petty criminals of the world.

So the question is: “Is there such a thing as a good virus?”

I asked this question to some of the viral experts.

Steve Dompier, programmer, evolutionary agent, and founder of Island Graphics Corporation, one of the fastest-growing computer companies in the country:

“One that kills the bad ones, obviously. What is really good about viruses is that people are beginning to create intelligent programs. I’ve been thinking about this ever since reading the publication of The Adolescence of P-1 (see Legends of the Cyberpast), a story written fifteen years ago about Computer Program #1, a self-aware piece of software.
"It only had a couple of rules. Its prime directive was to survive in any way it could. P-1 learned how to replicate itself, how to disguise itself, and how to jump through a computer network. It evolved to the point where it was controlling nuclear warfare games between Russia and the United States! Finally, the government killed it.

"Then one day, the guy who designed it walked by a teletype and it typed ‘P-1’ by itself. So he figured that P-1 had somehow gotten into the genetic code of the computer chips, and like the herpes virus, stayed in a dormant state for a while."

*John McAfee*, virus buster, president of the antiviral Interpath Corporation, and chairman of the Computer Virus Industry Association in Santa Clara, California:

"None that I can think of. None whatsoever. My mind has not yet conceived of a valid use for viruses except one—warfare. We could drop one off on the Chinese or the Russians or whoever. It could be timed to go off just before their next election, designed to be very, very slick so that no one could detect it.

"We could make it an embedded virus. It doesn’t change the date or time, doesn’t change the size, but it is there nonetheless. It can’t be seen and does absolutely nothing until it is time to destruct. Then on a certain date five years in advance, when our statisticians have figured out that every computer in Russia will be infected, it goes off and destroys everything.

"That is a truly valid use of a virus. Valid from some social standpoint in any case, from an offensive/defensive posture.

"But you’ve got to be damn sure you put some checks in that virus so that it does not infect any American computers. Because if it gets in one, the same thing could happen here. We could all go down together."

*Dr. Fred Cohen*, university professor, computer security specialist, and designer of many viral varieties:
"It's a more complicated issue than good and bad. I believe in knowledge and in attaining more of it. To the extent that we contemplate viruses and understand more about them, I think that is good. I think that there may be in the long term many uses for computer viruses that replace many of the things we now do with computers.

"I haven't written any papers on it yet, but I am examining things like viruses for distributed databases. I think I have a way of implementing a network of dissimilar computers operating a large distributed database without having a great deal of programming involved, with a very simple underlying mechanism, a very general means of spreading information and finding out information throughout the network. Viruses could be part of this information delivery system that automatically carries information from any computer system to any other."

Charles Wood, evolutionary San Francisco-based computer security consultant:

"Viruses could play an important role in an evolving immune system for our computers. They could be the equivalent of our white blood cells, traveling internal checker programs.

"Basically a decentralized defense from within, white blood cells and macrophages travel through our veins and arteries looking for 'unauthorized' visitors. In computers, roving 'checker programs' could travel paths leading to all the important parts of a system, disabling unauthorized or unrecognized programs, peripherals, terminals, and the like. These checker programs could refer to a log of authorized changes in production programs to determine whether a certain code should be permitted to execute or even whether it should be system-resident. It is in this way that unauthorized programs hidden within authorized programs (like viruses and Trojan horses) could be detected.

"Checker programs could also monitor the function-
ing of internal consistency checking mechanisms (parity checks, longitudinal redundancy checks, reasonableness checks between two data items in a database, and so forth). If a critical process were found to be amiss, rather than stop the process, the programs could trigger audible alarms and/or send notifying electronic mail messages. They could also terminate a process or isolate a user if required."

**Bill Me Tuesday, Bay Area high-tech hacker:**

"They can clean up the computer. And they can be used as a hacking tool in general. If someone is being stupid about their code [being secretive], they provide a good way of investigating closed systems. You can send viruses to all parts of a computer operating system to look for patterns as it is running. Viruses can act as a kind of a logic analyzer, if you like, and give you reports on how the system is performing."

These applications are just the tip of the iceberg. Considering the evidence, viruses are here to stay, and it would be best if we learn to harness their capabilities rather than fear them.
Who creates computer viruses? Are they criminals, heroes, or both? What role, if any, are they destined to play in our emerging world culture? And a world culture it is, for computer viruses, like radio waves or weather patterns, show no respect for humankind’s political and economic borders.

Viruses travel wherever they can, and are as playful or malevolent as the mind of their creator makes them. A well-written virus can infect thousands of machines an hour, obediently delivering its creator’s message and/or action to the unsuspecting computer user.

The early viruses had messages that were simple in scope and style. If they had a message, it was usually something like, “Ha! Ha! Ha! It’s off to Hell you go!” or simply, “Fuck You!”

**THE MORAL RESPONSE**

Arthur Abraham is president of A-Squared Systems, an Oakland-based company that creates desktop video
equipment for the Amiga computer. He has been a programmer for many years in Silicon Valley and has an inside track on what is new in the viral world. His thinking reflects that of many responsible programmers in this country:

"When one has been hacking for a while, it seems so simple to do all this weird, bizarre stuff. It is so easy to be totally nasty and destroy files and operating systems, or scribble your programs onto boot blocks of disks. But the thought that hits me right after those destructive thoughts is, 'No, that's wrong! That is not a good thing to do, and I won't do it! I will do something else equally clever, but perhaps useful instead.'

"That is the moral response to this kind of thing. A moral response should be instinctive in a well-civilized individual. But now there are so many people playing with computers that the technology is reaching people who aren't properly civilized.

"Infecting personal computers is like taking revenge against the world at large. It is like standing in a steeple and shooting at people in the street. Because so many people can be affected and the viruses are so undirected, the phenomenon is like atomic, chemical, or biological warfare. There are all kinds of people you don't want to hit, like your own guys. But because it is so personal, it can be one person against the world."

**BECOMING A VIRAL INVESTIGATOR**

The latest crop of viral designers deliver messages with more content. Witness the message that will appear on your screen should you become infected with the Brazilian Bug:

******* HELLO FRIEND *******

WE ARE SORRY TO INFORM YOU THAT YOUR SYSTEM IS VERY ILL AND WILL NOT SURVIVE MUCH LONGER
YOU UNFORTUNATELY HAVE
-THE BRAZILIAN BUG-
FOR WHICH THERE IS NO CURE
WE HOPE THAT THIS WILL NOT BE
AN INCONVENIENCE TO YOU
LOVE,
THE MAGIK MUSICIANS

After delivering this "friendly" message, your computer
begins to erase everything on your hard disk or floppies.
Definitely a malicious act, by any definition.

Can you imagine seeing this message, knowing that
the last six months of your life's work (like this book, in
my case) is on your hard disk, and that your backup
copies are most likely infected as well?

Technically, the Brazilian Bug is a very fast infector
and reinfector. And after reaching activation stage, it
does a low-level format of your drive. Its saving grace is
that it displays the message before it destroys your data,
giving you a little time to think about it. The message is
not displayed, however, until your system is well in­
fected (like finding out you have cancer with only six
minutes to live). Or, as John McAfee (its collector) notes,
"This virus doesn't surface until after it has done its
damage and there is nothing more you can do about it."

What would you do if a message of this sort suddenly
appeared on your screen? Most people I know would
probably stare dumbfoundedly at the screen, wondering
if it was all a joke. Then the next step would be to take
some kind of action to save the data, probably using one
of the many antiviral software treatments now available
(see Appendix C). Then, once a little security (and a sense
of humor) had been restored, you might wonder about
who created this little demon.

Reading the message carefully, one quickly observes
that the MAGIK MUSICIANS are a plurality, meaning
more than one. Hmmm... could there be a viral team
here? A gang, perhaps? If so, what other clues do we have that can help us identify them?

Well, let's look further at their name. MAGIK is spelled with a K, an affiliation with the darker side of the craft. Considering what they are doing, that fits. What is even more interesting is their chosen affiliation with music. What signs of musical talent are apparent in the virus? The timing of the message before complete destruction, perhaps? Maybe there is some sort of musical elegance in the viral code? Possible, but farfetched.

It appears that the Magik Musicians would like to be known for something, but are not. They can't really be known for writing viruses because they could be tossed into prison rather rapidly.

Psychologically, there is a certain thrill in doing something that everybody thinks is so bad, a thrill of recognition from others. Using a pseudonym of any sort is an indication that the viral designer wants to be discovered; his or her ego demands recognition. And the desire for credit can help get this person caught if you can figure out how to read between the lines.

**STATUS SEEKING AT ITS FINEST**

One of the world's experts in the human visual system is Dr. Robert Lambert. Blind since the age of five, he was raised in the slums of Philadelphia, Pennsylvania. He worked his way through college on a wrestling scholarship, earning his Ph.D. in mathematical psychology at the University of Pennsylvania, after his master's work in symbolic logic and the design of computer operating systems at Stanford University. Now a full professor of psychology at Concordia University in Montreal, Dr. Lambert has looked at the phenomenon of computer viruses from a psychological perspective:

"If you look at the Western system of perceived justice, there is a whole hierarchy of criminality that ranges from the basest of crimes like child abuse at the low end of the spectrum to relatively unimportant crimes, like pick-
pocketing or shoplifting. Then higher on the hierarchy are the white-collar crimes like embezzlement, and higher yet are high-tech crimes like those involving computer viruses.

"Everybody has contempt for the lowest forms of criminal, those who beat up or abuse kids. The criminals a little further up the line, the petty crooks and whatnot, people almost have sympathy for; people think of them as the poor woebegone of society. People think of the white-collar criminals as almost petty heroes. They were clever—the clever heroes, the clever con men; it's almost a shame they got caught. By the time you get up to the high-tech computer virus criminals, people tend to have out-and-out respect for them.

"They are status criminals. And as long as you are talking about a situation where you have status criminals, you are going to find people who are going to create viruses not only for reasons of revenge and retribution but simply as a way of attaining status. It's a good way for people to attain status when they haven't got any other way.

"I predict viruses are going to be a real problem. I think the world is full of people who are scared to death. There's very little success to be had out there. And everybody is running around trying to grab a little piece of success. Competition for success is so intense that, by and large, people will take it where they can get it. If they can grab a little success, status, recognition, a sense of their own value, their own self-worth—if they can grab a little of that by doing something illicit—they will. And why not? Especially if they can remain anonymous to the authorities. Writing computer viruses is a marvelous way of being a status criminal while maintaining anonymity. And you can do it from the safety of your living room."

Interesting perspective, eh? Now, let's take a visit to Hacker Haven, a top-secret location in the Santa Cruz mountains where hackers often come to gather their strength.
**A VISIT WITH THE HACKERS**

Hackers, for those unfamiliar with this subculture, are largely anarchistic computer-literate people who believe that access to information cannot be denied. Probably everyone in Western culture knows someone who belongs to this informal society, the person with the ham radio next door, the kid down the street who crashed his or her school's computer from home, the hacker in the office across the hall who's always tampering with everyone else's files. They all seem to be propelled by some inborn drive to do what few can do or have done.

They are the brethren of the high-tech frontier, the would-be merry pranksters of computerdom. The brethren often break new ground, thinking the unthinkable, charting the unknown. Wherever their minds go, we will all go eventually. Perhaps no group holds the future so much in their hands as the pioneers of today's super-technology.

It was at the Haven that I had an opportunity to speak to three members of the hacker community about computer viruses. Choosing to remain anonymous, they used the pseudonyms Ed Zackley, Bill Me Tuesday, and Dr. Expansion. Following hacker tradition, I did not interrupt the ongoing conversation of the group, but chose to join it at the appropriate moment. Bill Me Tuesday was speaking of the One-Time Pad Key System, a crypto-code currently used by the U.S. government:

**Bill Me Tuesday:** The One-Time Pad Key System is used for all embassy communications, communications to launch the missiles, wherever the utmost security is required. It is absolutely unbreakable, assuming your key is really random. The concept of the EXCLUSIVE OR gate is the basis of any crypto system. What makes it a one-time pad is that you use the key once and throw it away. No one has a chance to figure out the code. If you use it over and over, then patterns start to emerge.
The DES crypto-code, developed by the National Security Agency, was finally cracked by hackers. They used DES for the Videocipher system for TV satellites, scrambling HBO and Showtime. Hackers bastardized DES by playing repetitive information over it, which allowed the hackers to figure out the pattern. And that was it! Videocipher 1 and 2 are now completely broken. No one has to pay for satellite access to HBO or Showtime anymore. No one should have to pay for them except for the cable companies, and they don't need the scrambling anyway.

**Allan Lundell:** I had heard that many hackers consider viruses to be kind of a joke. What do you mean by that?

**BMT:** Just the fact that when you understand something, you don’t fear it. One of my latest projects is very software-intensive and could be attacked by viruses. I’m not afraid in the least. That’s because I’m including some hardware safeguards that will probably knock out any kind of virus.

A virus will never hurt a real hacker. Hackers who fear viruses have lost their hacker edge. Anyone should take precautions, but I don’t have any fear of the viruses.

**AL:** What’s your belief system? What would you like to see happen in this world?

**BMT:** If all of us had secure communications, where we can say what we want over the phone, the government wouldn’t be able to enforce their stupid laws anymore. People would be able to do more that the government doesn’t know about.

**Ed Zackley:** And vice versa.

**BMT:** The government is already doing a lot of things the people don’t know about because the government is heavily using security-oriented technology.

**EZ:** Well, the government is also being watched and listened in upon. They are. That’s a fact.

**BMT:** Usually by other governments.
EZ: Governments are more vulnerable than civilians by far, because the governments are in the public eye and people want to screw them up.

AL: But most people don't have the tools the government has.

EZ: True, but it doesn't take a lot of tools to reprogram their computers. That is one of the beauties of viruses. It's a David-and-Goliath kind of thing. A great equalizer.

BMT: I would like to communicate without being monitored.

EZ: I was ten or eleven when I first started using modems. I noticed people were really open about what they wanted on the bulletin boards. They were fearlessly making drug deals, exchanging products and information that were not government-sanctioned. But obviously these exchanges are not occurring today.

BMT: A personal one-time pad system would prevent all government monitoring of your communications, and a modification of DES would have them going for years. You could easily design your own unbreakable crypto-system for personal phone conversations with the new ISDN digital phones.

AL: Are you saying that it will soon be possible to have completely private, untappable phone conversations?

BMT: Absolutely!

AL: Earlier you mentioned viral telemetry as a good use for viruses. Could you elaborate a little on that idea?

BMT: One would insert a little piece of code that would tag modules in the program. The little piece of code would be a virus, looking for certain things to happen in those modules, then causing a branch or call when these certain events occurred. The virus would then let you know what those events were—like sentries that pass on to you information about what they are monitoring.

The virus would also call a subroutine simulating,
interrupting, and recording what all the status flags were and where the machine is at when something happened.

AL: So you could be looking for certain scenarios to occur...

BMT: You look for a certain pattern that would be helpful, and the virus would report on it.

AL: For personal use, you'd design the viruses to look for common problems that might occur in your software.

BMT: As the virus goes through the operating system, it stops at certain checkpoints, doing its rounds in a given amount of time. This checkpoint will report back what the condition is. Should a condition not be right, the virus will attempt to correct that situation. It will find the part that went bad, for example, and replace it. Essentially the virus will serve as a means of creating a self-repairing system.

The customer buys an infected system, infected with viruses designed for self-repair. They will also defend against invading viruses. If an alien virus tries to replicate in the system, it will alert one or more of the checkpoints. At that point, the invader will be removed by the system viruses. If viruses are going to attack my program, I'm going to write viruses that attack viruses. The goal is a self-repairing crash-resistant system, similar to the way our bodies repair themselves.

Biologically we are the product of thousands of microorganisms cooperating together. We can apply that kind of thinking in the computer world. We are modifying the concept of a virus to serve us.

Viruses could also eliminate computer bugs. Bugs often cause a piece of computer code to be stuck in a loop or run wild, tearing up data in the process. So if there is a very small piece of memory that keeps track of everything the computer does (like an aircraft flight recorder), one could backtrack when a piece of code goes crazy. A sentry virus monitoring
the situation could take action by snipping out that piece of code run amok and replace it with good code, and then one could put a program in there that would kill them all off when they were no longer needed.

AL: What is your experience of the political and emotional maturity of the hacker community? Do they have any real vision of what they want to create? Do they want a garden planet or what?

BMT: I see a strong ecological direction and also the elimination of greed. The end of the "me" generation. I think that is common. Greed just isn't allowed. He who gets greedy gets killed. Not killed physically, but financially. It's getting more and more that way. It makes me cringe a little bit to think about it. So when it gets out of hand, boom, they're had.

AL: There is no compunction about getting rid of greed-oriented institutions?

BMT: ... Or people.

AL: Do hackers consider themselves a counterforce to greed?

BMT: Yes, and it would be reasonable to assume that there are twenty thousand hackers out there and that, collectively, they have more power than all the governments.

AL: Through their ability to control the computer systems of the planet.

BMT: Hackers are anarchists. Governments are institutionalized, regimented, bureaucratic systems.

AL: Are there hackers working for the governments?

BMT: When they completely sell out, they will drop contact.

AL: With the hacker community?

BMT: Yes. Hackers can get in all the databases. Hackers know who is on the witness protection program. Hackers know who the Mafia are and so forth.

With all this knowledge, hackers don't let information go out to the wrong individuals. And basically, if
somebody does fink on a hacker, they very likely would end up dead or very harassed. I think dropping a dime on a hacker could be one of the most dangerous things someone could do.

AL: The community is that tight, eh?

BMT: I would say it is extremely tight. You wouldn't want to inform on a hacker and end up on the witness protection program. Some hackers have direct access to such information.

AL: So there are hackers who specialize in accessing certain kinds of information, and they make their data available to other hackers.

BMT: Yes. It is traded. It is very, very tight, you know. The Mafia could never gain access to real hackers because of their old-fashioned, worthless methods of doing things. If the Mafia want to use you, they want you for life. Because of that, no hacker is going to sell out to them. And if one did, that person would immediately be severed from the hacker community. That person would be useless and would probably end up getting offed by the Mafia.

AL: So there is no way of buying a hacker, because buying someone is a reflection of greed, and greed creates separation from the hacker community.

BMT: Right. You don't buy a hacker. You can hire a hacker, but you don't buy one. That very basic ethic is never violated, which is not to cross the whole setup. Take John Maxfield, who joined up with the FBI to inform on hackers. He knows all of us. Hackers have tapped his phone, run up his phone bill; he lives in hell. You just don't do what he did.

AL: Hackers are, by their very nature, anarchists and individualists.

BMT: I certainly agree anarchy can work. I support anarchy, and that does not mean no organization. Anarchy means organization without any long-term commitments. It gives me the freedom to change, to move on and check out other options all the time.

AL: With all this access, there are no real secrets, then,
provided you have the desire or need to know.

BMT: Hmm! That's an interesting way to look at it. Things coded Top Secret have a way of leaking out. Some are meant to, some aren't. Telemetry viruses will be useful for retrieving top-secret data. Information is power, and total information is total power; thank God, no one individual has total access to all the information.

AL: Or knows how to use it. You are saying that hacking drives technology and social values to be updated, eliminating the problems the outmoded technology and institutions are causing. A driving force.

BMT: Obviously. That's the way hacking always has worked and always will work. A pattern that works. And viruses are an example of this pattern. The law, for example, as an outmoded institution, resists change. The government takes the strictly legal approach to put people in prison, which doesn't work. It just causes anger and will help bring down society. Our society is getting lots of lawyers, lots of guppies, lots of useless service-oriented people out there. That's an indication of downfall. Study the fall of the Roman Empire.

AL: You are saying our culture has a lot of nonproductive jobs.

BMT: Yes. And the legal system is an example of nonproductivity.

DR. EXPANSION: The system is taking wealth rather than generating wealth. It also redistributes wealth. The fall of cultural structures reminds me of a phenomenon in virology called oncoviruses. When an organism reaches a certain level of deterioration or illness, viruses spontaneously appear in the system and help take it apart. They're agents of decay that serve to disassemble the system so that a new order can form. This is one of the many roles hackers seem to play with respect to social institutions such as the government or legal system.

BMT: Death viruses. Oh! If you think about how govern-
ments and societies at large depend on computers, hackers have taken over the world. No individual could take over the world; Hitler came pretty close, but he blew it because of his blown-out ego.

AL: You say hackers have taken over the world?
BMT: Hackers have the world. As things get out of hand, there are several thousand hackers out there who will take action.

THE IDEOLOGICAL MOTIVE

Researching this chapter has given me access to some amazing individuals, most of whom have chosen to remain anonymous. There is no one common denominator among them. Collectively these people represent a wide cross-section of humanity, interested in everything from computer hacking to international banking.

The individuals I found most interesting were those with ideals that they felt computer viruses could help uphold. "St. Paddy" is one of those persons.

After migrating from Detroit to the Bay Area in 1980, he started a nutritional supplements company that is now a multimillion-dollar business. His Irish Catholic upbringing in the slums of Detroit obviously helped shape his feisty nature, bringing him to a highly animated state during parts of our interview.

Now semiretired at forty, and living in his beautiful garden home in the Santa Cruz mountains, he has set himself up with a state-of-the-art video/computer studio for purposes of disseminating important information he feels is being kept from the American public.

A mover and shaker, he has many planned uses for viruses and has befriended several key viral hackers because of his spiritual/political beliefs. Like Burleson, he dislikes the IRS and the present economic/power system. After you have read the following interview, you will see why I have kept his identity a secret.

Allan Lundell: What do you wish to achieve through the use of computer viruses?
**ST. PADDY:** My basic goal is to get Big Brother off our backs and make the planet a safe place, eliminating nuclear warfare. One of the ways of eliminating nuclear warfare is to disconnect the defense computers around the world.

**AL:** Don't you think the powers that be will want to get rid of you for this kind of thinking?

**ST. P:** Of course, but I am an everyday Joe in most respects. Millions of people have the ideals to change the power structure of the planet and end the threat of nuclear devastation. And these people don't belong to any particular group. Each one is a unique individual without any common religious background.

**AL:** What is driving you to be a revolutionary today?

**ST. P:** I have really studied the banking laws of this country, and I have found that these laws are not in support of American citizens at all. They are designed solely to support a few elite individuals in our society, and these individuals also control most foreign governments.

In this country most senators were bribed and misled to vote favorably for the legislation leading to creation of the Federal Reserve Bank.

The Constitution says only the Congress shall have the right to coin money. It does not say that the right can be delegated to someone else. When they enacted the Federal Reserve System, the FRS was not a branch of the government. It still isn't. It is an independent bank!

What the U.S. Government did was give the most powerful bankers in the world a complete monopoly on the whole economic system for one purpose—their own profits!

In simplified terms, here's what happens when the U.S. Government wants money: they issue U.S. Bonds. Say they need a billion dollars. They issue a billion dollars' worth of bonds. They give that to the Federal Reserve System. The FRS calls up the U.S.
Mint, Engraving and Printing Department, which is run by the government. Run at taxpayers' expense! They call them up and say, "Print a billion dollars," in whatever denominations they want.

Fine. It's done. They deliver the money to the FRS. The FRS writes that off as a loan to the U.S. Government and charges them interest!

They've created the money out of thin air at taxpayers' expense. They then charge the taxpayers of the United States interest on that, and then they sell the bonds and collect interest on that too!

AL: It's the bankers who own the Federal Reserve Bank?
Sr. P: Yes! There are twelve branches, like the Federal Reserve Bank of New York, Cincinnati, Houston, San Francisco. And all the banks in this country get their money from the Federal Reserve System. So in that way they control all the banks in the country, either directly or indirectly. Even though most of the banks are privately owned, they are charged interest by the Federal Reserve System. And restricted through national banking laws and regulations through the all-powerful influence of the federal and world bankers.

AL: So it is not the U.S. Government running the Federal Reserve System?
Sr. P: That's right! In real life, at all levels of society, a man is a slave to his creditor. The bottom line is that you are a slave of the one to whom you owe money—including your landlord and the bank that has a mortgage on your house and the deed to your car. If you don't make your car payments, they take your car away! If you don't pay your electrical bill, they turn off your electricity!

AL: You are a slave in the sense that you have to figure out how to get the money to pay your creditors, and it takes your personal time and effort to do that.

Sr. P: This same process happens at higher levels of government, too. When a country is indebted to the bankers to the tune of $3 trillion, you don't think the boys' club tells the president what to do?
AL: To get the money?
Sr. P: Yes. Or the FRS will cut off the money and wreak havoc with the economy. If Ronald Reagan had not done what they wanted him to do, he would have been out in 1984, because he made such a bad economy.
AL: Why doesn't the United States print more money and pay the bills?
Sr. P: Because the bankers own the money, and they would just charge the United States more interest. It's this great scam they've got going. The only way they are going to pay back the bills is to get the American taxpayers to pay more taxes.
AL: So that is the other source of income for the United States—taxpayer dollars. Income sources are printing money out of thin air from the FRS and U.S. income tax.
Sr. P: That's right! Any payments the United States pays back to the Federal Reserve on our loan to reduce the debt, is done through American taxpayers! Not only that, when the World Bank or the Export-Import Bank made all the loans to the Third World countries, they have never been paid back. Do you know where they will get the money? From the American taxpayer!
In 1985 Reagan wrote a check to the World Bank for something like $17 billion. Where did he get that money? From the U.S. taxpayers!
The Federal Reserve System is a complete sham. Our government leaders sold the American public completely down the tubes. The country was sold. The banks literally own the country right now.
AL: What is their ultimate goal?
Sr. P: I believe they want to own the entire world. The ultimate power trip is monopoly of the world, to own all the world's resources and labor.
AL: What about the Communist world?
Sr. P: Moscow is run out of New York City. Another word for communism is monopoly. Who owns Russia? The
state owns it. Now you must ask yourself one more question: Who owns the state? Who are they indebted to? From where did they get their money? Who paid for their revolution? I believe the bankers did.

The bankers own everything, and their ultimate goal is the formation of a one-world government. They want all of the wealth of the world in their hands.

AL: What do you do with this information?
ST. P: Get it out there to the public. Show people how the bankers are taking their Constitution away from them. And computer viruses are a means of distributing information quickly throughout the computer networks without censorship, a means of breaking through the disinformation disseminated through the traditional media.

AL: Basically, what is your personal belief system?
ST. P: We are spirits trapped in a physical body. Part of my belief system is that part of being in a physical plane is to experience imperfection.

AL: Isn’t part of it also to bring mastery from other levels of the universe into the physical?
ST. P: Yes, that is the game. Eventually we evolve from imperfection into perfection, but we are not there yet. We are not even close to it yet.

AL: Is God perfecting Himself in the physical domain, the hardest level of the universe to completely master?
ST. P: That would be my theory. I agree with that. Perfecting the densest of reality, the last step of evolution. I think we need to set up massive education programs, workfare programs, so people can learn how to evolve, to be able to see who they truly are.

AL: Would you say that one of the first things to do is to create time and space for massive education, where the populace is not slaving away for the banks? Once survival issues are handled, the natural direction of
the species is toward personal perfection?
St. P: Essentially, yes!

VIRUSES, MEMES, EVOLUTION, AND GOD

Dr. Fred Cohen, creator of the first truly replicating virus, is a major player in the viral community. He feels God has an intimate knowledge of viruses: "If you think of the whole concept of God, you are talking about a virus.

"Richard Dawkin in *The Selfish Gene* [Oxford Press, 1978] talks about genes and says that the mental versions of genes are memes. Memes are the transmission of ideas from person to person, culture to culture. He says that memes last longer than genes. By the time you get to five or six generations in genetics, you lose virtually all the information about the individual. Whereas with memes, you look at the efforts of, say, Aristotle done hundreds of generations ago, and we still know Aristotle and his work. The concepts of Aristotle's we learn today are memes—mental genes, if you wish."

AL: Dead men do live—via the lips of living men. Something that we consciously pass on, it seems.

DR. FRED COHEN: It doesn't have to be conscious. It doesn't have to be purposeful to be passed on.

AL: So memes are a form of nonbiological genetics . . . mental, cognitive . . .

FRED: That's right. And people thinking and communicating are just like computers computing and communicating. Our modern computers, of course, aren't up to what people can do with their minds, but at a very deep theoretical level, there is almost no distinction.

The same is true of societies. Look at the way newspapers work, for example. You may get some minor corruption—the president says to a couple of newspaper reporters that Qaddafi ordered a discotheque bombed, and all of a sudden, every newspa-
per in the United States has that corrupt information, and in fact years later, you may find many, many people who believe that this was the case. They have never heard the other side. This is an example of a "transitive integrity corruption," which is in every way a mental version of a virus.

An interesting analogy is the party game Telephone. In this game, a group of people sit around a table, and one person whispers a message to the next, and so the message goes around the table, eventually coming back to the first person. If you thing about mental viruses and the Telephone game, you are passing communications along, and the message evolves. That's evolution, right?

AL: Yeah, that's it. And sometimes the message or story that comes around is completely different from the original version, unless there is some sort of error checking of the data as the message is being passed.

FRED: Unless there is some form of redundancy (like getting the person to whom you told the message to repeat what you just told him or her). In writing the theory of evolution, we as scientists try to make the theory cover all possible types of evolution, so that when we talk about evolution, we have a strict definition that covers everything we normally associate with evolution. One of the interesting results I have achieved with this kind of thinking is a virus that can evolve into a form that solves any problem a computer can compute. So anything that can be solved by a computer performing computations can be solved by an evolving virus.

**RADIO HYPNOSIS:**

**A MIND VIRUS IN ACTION**

One night while in college, I played "The Great Pecarve Show" over the radio. Pecarve was a hypnotist who did the college circuit with his act. As a producer at the campus radio station, I recorded his show and inter-
viewed him. He was an interesting guest and demonstrated the power of hypnosis profoundly. Then one night I had an idea. What if I used his hypnotic induction routine to experiment with mass hypnosis over the radio?

The station was broadcasting only on cable FM. We were known for unusual and bizarre shows, and, besides, it seemed to us that anybody who liked our programming deserved to be hypnotized. What’s the worst that could happen, anyway?

Anyone listening to my show that night heard something like this:

You are getting sleepier and sleepier. As I speak, you are going deeper and deeper to sleep. More and more relaxed, deeper and deeper to sleep. . . . Now, you hear me, but you are deeply asleep, deeply asleep. And now you will listen carefully. As I count from five to one, you will awaken. But when you awaken, you will not remember your name! You will not know who you are. You are truly a mystery to yourself. And upon awakening, you will have a burning desire to phone our station, asking us who you are. Five, four, three, two, one—you are awake!

At this point, the radio station’s five-line phone lit up like a busy switchboard. Rather than answering the phone, I used a hacker invention. I connected all five lines together, allowing the five parties calling in to speak with each other. Listening to them trying to figure out what was going on and who they thought they were was a real-time lesson in the psychology of mass hypnosis. Their conversation sounded something like this:

"Who are you?"
"I don’t know! Who are you?"
"Why am I talking to you?"
"I don’t know. What is going on here?"
"Beats me."
"Hello? Who is this?"
In hindsight, this college radio experiment had a viral quality. It was about how thoughts take over a host and how these new thoughts can affect and infect people's behavior. In this case, the hosts were the audience, and I was the originator of the thoughts—the mind virus.

My on-air suggestions had taken over the mental operating system of many of my listeners. They did what was suggested rather than what they would normally do. And what were suggested were things they would probably never have done by themselves. Some of the callers were so deeply hypnotized that I could have told them to come to the station, and they would have done so. They had turned their systems over to me.

Interesting experiment, eh? Well, my major was human experimental psychology. I did return all the listeners back to their own operating system later by inducing a trance state and saying on air, "All of you now remember who you are and what you are doing. And you just had a great time learning about yourself and hypnosis."

TECHNOCHASM: EVOLUTION IN ACTION

There is a major chasm between those who are running this culture and those who actually understand and create the technology. There has not been a science adviser in the White House since Carter's administration, and not one taken seriously since Kennedy's.

Those who are creating the technology today are more sophisticated than previous generations. With more leisure time to philosophize about who they are and what they are doing here, the New Age and "me" generations of the baby boomers are reflections of a desire for increased self-awareness.

The personal computer facilitated this revolution, as it has increased worldwide communications, giving more of us instant access to what is going on globally. The world gets smaller every day.

In our parents' time, great inventions, like the atomic bomb or television, were often the domain of large corpo-
rations or the government. These inventions were so expensive that only wealthy groups or individuals could afford to maintain them.

Those in charge of the atom bomb, for instance, were removed from the direct experience and understanding of its power. It wasn't Roosevelt who saw the forces of creation and destruction literally dance before his eyes in the New Mexico desert. Dr. Robert Oppenheimer, the human source of the new power that could transform the earth once again into a garden planet was the source who would later be denied access to the invention he helped create, because of differences in political and ideological belief.

Today, equally powerful ideas and inventions are being unleashed, but these ideas are more accessible to the public at large. Video technology, microcomputers, home satellite technology, worldwide telecommunications, electronic libraries, and soon, home VTOL aircraft are current examples.

Today, there are more inventors than ever before and more means to make their inventions real within our culture. This is a true redistribution of power. No one group decides what is created. Now an idea can be taken from paper to product in a few months, and new power is being evolved from new inventions by the public itself. With the advent of computer viruses, over a million computer-literate Americans have been invested with power over the institutions that serve or hinder their life experiences.

Now the very survival of these institutions is at risk, and it is my guess that the institutions surviving this viral invasion will be those that best serve the interests of the many.
"The most important quality of biology lies in its ability to increase numbers explosively. During the interval of two to three days between 'catching' a cold and the moment when you first begin to sneeze violently, the common cold virus multiplies its number inside you about ten thousand millionfold. And if the virus could keep on growing at the same rate for three weeks instead of three days, the eventual mass of it would exceed the combined masses of all the stars in the Milky Way."

—Sir Fred Hoyle and Prof. Chandra Wickramasinghe, Diseases from Space

**BIOLOGY vs. SOFTWARE**

The biological viruses are amazing little creatures in their ability to replicate. Given half a chance, they would very quickly kill our species in favor of their own reproductive goals.

Nature, however, has struck a balance between their reproductive zeal and our survival. Defending ourselves
against viruses has made us a very disease-resistant life-form, with an immune system capable of destroying hordes of these virulent microscopic invaders.

There are a few key similarities and differences between the carbon-based biological viruses and the silicon-based computer viruses.

Biological viruses are like slave drivers from hell. They enter our cells and take over the operation of the cell’s brain, forcing it to do the virus’s bidding. The infected cellular brain, or nucleus, is forced to issue commands to the rest of the cell, commands that tell the cell to manufacture viruses rather than producing sustenance for the cell.

In a very short period of time, the infected cell is manufacturing thousands of viruses. As the manufacturing supplies run out, the cell dies, and the thousands of new viruses are released to seek out other cells to invade. This goes on until either the viruses or the cells are destroyed.

Computer viruses enter the program they are infecting, installing themselves in a location where they can take control of the action commands issued to the program from the computer’s operating system. The infected program then remains inactive as the commands meant for it are intercepted and used by the virus for replicating itself. The virus usually makes only two or three copies of itself after each system command. It returns control to its host program, which then boots up and runs normally. From the computer user’s perspective, the only symptoms of this process are slight delays in the booting up of infecting software, usually up to a second or so.

The biological viruses specialize. They choose only certain types of human cells, like central nervous system neural ganglion cells or kidney cells.

Computer viruses also specialize. They infect only program files that will allow them to replicate. Specialization is a means of survival.

How do viruses know exactly how to take over and use
the power of the nucleus, or how to specialize? They obviously have an intimate knowledge of how our cells work, and this knowledge about our cells is encoded in their operating system, the viral code.

In many ways, the ability of the biovirus to manipulate cellular function exceeds what our science knows about cellular control mechanisms. For example, we don’t really know how to command a cell to create viruses.

By studying the viral code, we may learn to read it as a blueprint of how to manipulate the function of our own cells. Through genetic engineering, perhaps we may someday be able to create or modify viruses that could improve, rather than destroy, cellular function. The act of creating viruses as a means of influencing cellular activity is state-of-the-art thinking in biological circles.

By comparison, creating viruses is rapidly becoming commonplace in computer circles. This is because computer viruses are far simpler creatures than biological viruses, and require far less technology to create. Hence, they are easier to manipulate.

To create a computer virus, all one needs is a personal computer and some knowledge of software programming. Computer designers know exactly how a virus works and how to program one. Provided one has the interest, how computer viruses work is no mystery.

To create a biological virus, all that’s needed is a gene-splicing laboratory and a respectable understanding of the DNA code. Biological scientists don’t know exactly how a virus works or exactly how to program it. They are in a process of figuring out how viruses work, and that process is very much like unraveling a mystery.

In both the biological and computer arenas, one can have more influence on viral activity by knowing how to design and redesign viruses than by simply defending oneself against the creatures. Know your enemies by putting yourself in their shoes and in that way, you will know who they are and what you are dealing with.

Biological viruses learn through trial and error. Those that replicate continue to live. Those that don’t work,
don't replicate. It's a survival game won by the most functional.

Computer viruses don't learn through trial and error or by any other means. Those viruses in the present generation are simply bits of programming code, too stupid to do much but hide, replicate, and then fire off a bomb at a predetermined time.

Biological viruses don't have warheads installed by their creator. They simply replicate, destroying their hosts by depleting cellular and immune system resources.

Computer viruses often have a warhead, designed by their human creator to go off at a certain time or when certain conditions have been fulfilled.

Unlike the biological viruses, computer viruses require a human creator whose excitement and enthusiasm are born of the perception of his or her own capacity. When this impetus of enthusiasm and excitement fades away (no more coverage on the six o'clock news), the viruses lose their vitality, unable to reproduce themselves, they lose their ability to preempt the "genetic material."

Biological viruses do not go through such cycles of excitement and dormancy. Their functionality is not dependent on the human psyche.

Biological viruses are agents of our destruction, and we are agents of their destruction.

Computer viruses are agents of the human mind, leading to whatever ends the human mind can concoct for them.

The patterns by which biological viruses spread are either endemic or epidemic. That is, viruses are either constantly present and infect only a few people at a time, or they come and go in waves, attacking whole populations, then fading away.

Computer viruses so far have been epidemic, though with the development of new shield-type antiviral products, endemic patterns of spread will undoubtedly evolve.
BIological processes are now being studied by the computer security community as a means of improving security methods and as a way of stimulating new thinking in the field. Charles Wood, a San Francisco security consultant, recently wrote a brilliant article on the human immune system as a reference model for creating a secure information system. A reference model refers to a way of thinking, a map you can consult when organizing your thoughts.

Until now, most computer security reference models have been based on the military model. An example of thinking via the military reference model is the concept of password access. Password access to a computer is based on the analogy of a fence or wall protecting a valued physical asset. When you give the correct password you are allowed through the electronic logical fence around the computer.

MILITARY SECURITY REFERENCE MODEL

In the military reference model, system penetrators are considered hostile invaders. Wood points out that this view is not always accurate; often "hostile system penetrators" are only curious employees or exploring high-school students.

"For many years, work in the field of information systems security has been a reaction to our fear of such events as service interruption, industrial espionage, privacy violation, sabotage, and fraud. We find we are repeatedly asking ourselves questions like, 'What if this went wrong?' and 'What if that were to be compromised?' Consider moving beyond this reactionary and narrowly defined way of thinking to thinking about the creation of a vastly different and new environment made possible by computer and data communications systems."
And what might this new environment be like?

"This could be an environment where people are supported in their efforts to get the job done efficiently and effectively, expand their capabilities, cultivate their creativity, and otherwise fulfill their human potential. Control measures would actively assist users in properly protecting their information assets and would automatically protect users from damaging their data. This new way of thinking creatively about information systems can give us true dominion over the technology; the traditional way of thinking only reinforces our notion that we are victims of events beyond our control."

And what exactly are some of Wood's evolutionary ideas? Using the human body with primary focus on the immune system as a reference model, Wood outlines seven different principles that can be applied to computer security:

**Adaptation: Robust Systems That Adapt to Change**

Analogous to human muscles becoming conditioned to handle hard physical work, machines will be able to change in response to external stimuli. More specifically, disaster recovery could be enhanced by software that automatically adapts and responds in the moment to the situation at hand, like bringing on line appropriate back-ups, a new host machine, new peripherals, or a new set of application priorities. Also, the defenses of a system could change in response to certain types of attack.

**Vaccinations: Automatic Testing Procedures**

Vaccinations can be viewed as small stresses to the body that prepare it to deal with greater stresses. By injecting less virulent viruses, the body generates antibodies, which in turn attack the active virus when it enters the body. A computer parallel could involve hackers who mostly engage in relatively innocuous activities that identify and encourage the system to correct
security deficiencies. Antibody software could be developed that counters specific types of attacks to the computer system (such as guessing log-on passwords). This analogy can be taken even further by developing permanent system-resident auditing software that would continuously generate new system tests (always developing new antibodies). These programs would decide what mechanisms and functions of the system need testing and protection next.

*White Blood Cells: Traveling Internal Checker Programs*

Basically a decentralized defense from within, white blood cells and macrophages travel through our veins and arteries looking for "unauthorized" visitors. In computers, roving "checker programs" could travel paths leading to all important parts of a system, disabling unauthorized or unrecognized programs, peripherals, terminals, and the like. These checker programs could refer to a log of authorized production program changes to determine whether a certain code should be permitted to execute or even whether it should be system-resident. In this way, unauthorized programs hidden within authorized programs (like viruses and Trojan horses) could be detected.

Checker programs could also monitor the functioning of mechanisms for checking internal consistency (parity checks, longitudinal redundancy checks, reasonableness checks between two data items in a database, and so forth). If a critical process were found to be amiss, rather than stop the process, the programs could trigger audible alarms or send notifying electronic mail messages. They could also terminate a process or isolate a user if required.

Isolation is a powerful means of stopping infection from spreading to other parts of a human body and can be an approach to limiting the damage done by unauthorized computer users, systems, or processes (like viruses).
Antigens

Cancer cells have telltale marks on their surfaces by which the body identifies them as foreign. These marks are called antigens and play an important role in the development of vaccines that provide an immunity against certain types of cancer. In computer systems, antigens are comparable to specific sequences of bits, bytes, characters, or other representations of information. With intelligent logging systems, computers will be able to recognize automatically the correspondence between a current attack or breakdown and a past attack or breakdown. Following detection of a correspondence, appropriate action could be taken in real time to intervene automatically. In other words, whatever worked in the past could be instantly applied in the present.

One of the issues here, of course, is storing and accessing humongous amounts of data. It is possible that the massive quantities of data required for these comparisons with past attacks and breakdowns could be stored on optical storage devices. Such capabilities to recognize patterns could assist in the prevention of system interruptions in service, corruption of data, and unauthorized use and abuse.

Free Radicals

Free radicals are chemically reactive entities in the human body created by exposure to radiation, the breakdown of rancid fats, normal metabolism, and other activities. These chemicals can damage the body by cross-linking (the progressive formation of chemical bonds or bridges between large molecules, such as proteins and nucleic acids).

A parallel in the computer environment is errors and omissions (E&Os). E&Os can cause cascade or domino-type effects in which the modification of data entities leads to the inappropriate modification of many other related entities.
A control acting as a free-radical (E&O) neutralizer would be of considerable value in our increasingly complex, integrated, and high-speed computer systems. These free-radical neutralizers could detect E&Os that applications or other program code had missed, particularly those E&Os involving the interaction between several programs, devices, computers, users, or some combination of these.

These controls could examine all processes for efficiency and relevance to current operations. Processes that were resource "hogs" or that were unnecessary would be placed in archival storage, replaced with an optimized version, or both.

**Inflammation and Fever**

Inflammation in the body is generally caused by a histamine reaction, the same substance that causes allergy symptoms. Usually this inflammation leads to an expansion of the size of blood vessels at the site of the problem. This blood vessel expansion allows more blood to be pumped to the site and also allows the blood vessels to become more permeable, so that certain defense substances (like antibody assistors) are left at the site.

In responding to problems, computers could alter themselves in real time to bring in checker programs, slowing down or stopping ordinary work until the problem was rectified. Instead of continuing as though nothing were the matter, a fault-tolerant computer could hand off production processing to another processor while repairs were being made automatically at the affected processor. Once the ailing processor was fixed and tested, it could be brought back to active status automatically. Self-repairing systems could become the new norm.

This approach could significantly reduce the probability that errors and omissions, fraud, and other problems
went unnoticed, thereby avoiding erroneous management decisions, inappropriate computer control of human-life-threatening machines (such as aircraft navigational aids), and other serious consequences.

THE REPRODUCTION CYCLE OF THE AIDS BIOVIRUS

Computer AIDS
(Acquired Immuno-Deficiency Syndrome)

Not all aspects of the immune analogy are beneficial to the health of the computer system. It is possible, for example, to develop a form of computer AIDS. Computer processes could be brought to an unanticipated and untimely halt or seriously damaged by the very controls that were intended to protect them. This is true for any model. In the present systems, for example, losing or forgetting your password can cause major problems.

In our model of the computer immune system, for
example, the virulence of a virus can crash its host or hosts very quickly. An attack on the computer's immune system could occur if a white blood cell checker program attacked other checker programs, should its integrity be infiltrated by a virus. The system could also crash should antibody-related destroyer programs attack legitimate processes, or if a control infinitely recursively called itself.

So how is it possible to trust new, powerful control programs? How can we make sure they are functioning properly? How can we be sure that they know the differences (and similarities) between invaders and what they are supposed to be protecting? Studying the human immune system may provide answers to these questions.

Recent AIDS research, for example, has discovered a nucleotide analog that fools the AIDS virus into linking up with it rather than DNA, thereby stopping further reproduction of the AIDS virus. In a computer system, a virus program might be fooled into believing that it was indeed inside a computer system, or that it had certain privileges, when such was not the case. The virus program's destructive actions would then be revealed and give the controlling software reason to destroy it.

Wood's analogy between system security and the human body's immune system is in alignment with the present shift in our cultural paradigm toward holistic thinking, where the planet is perceived as a single living organism with its intricately linked webs of networks and systems. This kind of thinking leads toward the perception of self as part of a much larger organism, where we in relation to the planet are like viruses in relation to our bodies. Such thinking also sparks much creativity and, among other things, leads to the development of self-thinking and -repairing robots and androids, capable of sustaining themselves, in addition to being compatible with and supportive of their environment.

It is also useful to think of the body's systems as analogous to social structures in our control. Like computer security, country and ultimately species security
can be co-related with our personal immune systems. The computers, an externalization of our mental faculties, are now establishing the validity of this new kind of thinking.

We leave you with some basic biological virus data:

- The most contagious virus is measles.
- The most deadly virus is rabies.
- The mildest virus is warts.
- The most mysterious disease is virally spread cancer.
"I think there is only one real way you are going to cure the virus problem. Basically, we’ve got to have lots of them. When we have twenty million people who are capable of developing viruses and not one million, it will become child’s play. Then, there will no longer be any merit or status in making them."

—Dr. Robert Lambert, Computer Psychologist, Concordia University

As of this writing, there are no foolproof means of eliminating computer viruses. There are vaccines, antidotes, detection methods, and preventive measures that can be taken, but no guaranteed cure.

The reasons for this sad state of affairs are numerous. Finding a computer virus inside a software program is about as difficult as locating a needle in a haystack or a piece of cellophane tape hidden somewhere in your home. It could be attached to the back of the sofa, under the dining room table, inside a cabinet, or under the bed.

“Well,” you might muse, “at least we know to look for
cellophane tape. It might take all day, but sooner or later we will find it!"

The real situation is not so simple. The more clever viruses can have cloaking abilities masquerading as something else. There are also tremendous variations among viruses, making it extremely difficult to know what to search for.

A popular approach to stalking viruses involves the use of a reference piece of software. If you can compare your software with what you know to be an uncontaminated version of it, you can detect any differences that may be present. This technique works well, provided you have a way of maintaining an unadulterated, pristine copy of your software and an antiviral program that can do the line-by-line comparison without taking forever.

In June 1988, John McAfee started a virus trade association to set guidelines for the development and marketing of antiviral measures. As in the early days of human medicine, we can expect to be offered some "snake oil" cures as computer viruses continue to proliferate.

"Computer viruses and antiviral measures have become a rapidly growing, changing, and increasingly complex field," states McAfee. "Limited access to controlled viruses and lack of understanding on the part of the user have created an environment conducive to misinformation, emotionalism, and fraud. If the industry does not quickly police itself, the proliferation of unverified and untested products may cause irreparable damage."

Idealistically stated, but the road to success for his organization has been hard for McAfee. Accused of activities like seeding real viruses into the corporate world to generate business for himself or of using the virus press coverage he has received for pushing his own products, he is a less than popular man with many of the antiviral companies.

His plans for creating an unbiased test bed for antiviral software are about as acceptable as having only one pharmaceutical company running the Center for Disease Control, setting the standards of new drug testing for all
the products of other pharmaceutical companies. McAfee's ideas are great, but the reality of the situation cries for an unbiased entity like the Center for Disease Control, *Consumer Reports*, or Underwriters Laboratory. In defense of his position, McAfee says, "Our association is not going to go out and rate people and say this is good and this is bad. What we are doing is to say, 'Look, here is what we think is a legitimate test of antivirus systems. We will submit your products to this test.'"

A key point of disagreement lies in what exactly constitutes a legitimate antiviral test. A legitimate test can be defined only in terms of some type of industry standard, and to have any real meaning, this industry standard must be accepted by all players in the industry.

So far, most of the industry has agreed to the following basic classification of antiviral products:

- **Class One**—Products that prevent viral infections from getting into your system. Basically, they are high-tech computer prophylactics.

- **Class Two**—Products that detect infection within your system soon after it has occurred. These products also mark the specific components of segments of the system that have become infected.

- **Class Three**—Products that identify specific viral strains on systems that are already infected. These products are also capable of actually removing the virus from the infected computer system.

Most computer vaccines are preventive measures and fit in the Class One category. They take advantage of the fact that the viruses often hide themselves in specific locations within the computer's control software. A good vaccine will stake out these hideouts, watching and waiting for the intruders to arrive. When one does arrive, the vaccine will stop computer activity and alert the computer operator that one of the hideouts is being occupied.
Vaccines are limited in that they can’t monitor all possible viral hideouts or even necessarily ascertain that the activity occurring in the hideout is illegal. Furthermore, new viral strains don’t always use the same hideouts, and some can masquerade as normal computer activity. In other words, vaccines can be bypassed and work best in detecting viruses that have already been caught and identified.

Once a computer has been infected by a virus, the invader can be caught by a Class Two product. McAfee’s program Tracer, for example, identifies the segments within the program that have been infected. It collects all the interrupt vectors, the favorite 130 or so cells viruses like to use, and reads the boot sector in its entirety.

Tracer also reads all the executable programs, all the system device drivers, CONFIG.SYS, and a whole host of other things. It logs the date, the time of creation, and the size, and also, to trap viruses that actually embed themselves within programs, it reads the first fifty bytes of the initial load segment. This will trap at least 99 percent of all viruses, according to McAfee.

It is possible to design a virus that would bypass Tracer. (It would go in and pick up a branch instruction in the middle of a program and then go to a data area.) But it would have to be a specific virus, not general-purpose. In other words, it would be tough to design a virus that could infect any program in such a fashion.

The disadvantage of a program like Tracer over a vaccine is that, while Tracer is 99 percent effective, your system has to be infected before it can be used. Vaccines prevent an infection but can be bypassed.

One of the best examples of a Class Three product is KillScores, an antiviral formula developed by the MacPac AppleCore user group of Dallas, and put into widespread distribution by Howard Upchurch.

KillScores is designed to repair any Macintosh system infected with Scores. When KillScores encounters an infected program, it tells the user which file, and makes a recommendation. As a user, you can ignore it, delete it,
or repair it. If you follow the automatic recommendation, and if you do have the Scores virus, your hard disk will be fixed in about ten minutes.

**THE MAC APPROACH TO VIRAL CURES**

The viral trade association started by John McAfee represents products that are for the IBM PC and PC-compatible world. The members of the association are companies that have antiviral products to sell. It is a competitive environment, reflecting business issues. Products are promoted, packaged, marketed, and distributed. This process, of course, takes time, a commodity that is at a premium in the fast-mutating world of computer viruses.

Owners of Apple Macintoshes are more community-oriented, choosing to exchange information with each other over computer networks and through computer user groups. When the nVIR and Scores viruses hit the community, the networks and user groups literally buzzed with viral information. Everyone who cared knew who was hit and how badly they were affected by the viruses. The community took it upon themselves to find cures for their ills, and within a month of the initial nVIR epidemic, several antiviral cures were circulating among the user groups. Here's how Howard Upchurch described the way KillScores was first distributed in Dallas:

"I received it from the programmer on a Sunday morning. It did not even have an icon, just the program. Well, we had a mad rush that day, as the user club meeting was that Sunday evening. We designed an icon, put all the antiviral stuff we had on one disk, and started making copies. That night we sold 120 disks with KillScores for $2 each, barely covering our costs."

Besides the cures, of which there are about half a dozen for the Scores virus alone, there are stacks of information about the viruses. Dr. John Norstaad, at Northwestern University, even assembled a hypercard stack detailing the purpose of the virus! (A hypercard
In the words of Don Brown of CE Software, whose efforts to stop the Scores virus have been a beacon for the whole Mac community, "The whole thrust of the personal computer has been bringing control of the computer to the user. Viruses steal that control away and replace it with fear, uncertainty, and doubt. Why would anyone want to take such a gigantic step backward?"

**THE APPLE CORPORATE RESPONSE**

Apple Computer, however, was a bit more conservative in its approach to viruses. The company formed a secret team, known as the Immune Group, to investigate the viral threat. Apple had experienced some internal viral problems, including an unexplained failure of its voice mail system and numerous infections by the Scores virus of machines within the corporate campus. The loaning of infected Macs to outside businesses, without realizing what they were doing was very embarrassing for Apple.

Apple’s Immune Group has discovered some new viruses, ones that look potentially ominous. Apple likes to keep their work secret, and in respecting their wishes, I leave the source of the following description of the Invisibility virus anonymous:

"I ran across a virus that was particularly difficult to get rid of because the virus made itself invisible as soon as it knew that the vaccine Interferon was introduced to the system. The virus made the file invisible but attached itself to the file. When I went to find it, I couldn’t. When I tried to eradicate it with Interferon it would put Interferon into a loop that would not allow the virus to go through the DELETE command, I could get rid of it by deleting it via ResEdit, as the editor renders it visible. But like most Mac users, if I didn’t know how to use ResEdit, I’d have a tough time getting rid of this one. Sometimes people can destroy the entire disk trying to use ResEdit. If left in the system, the virus would end up infecting
other programs. The virus exists physically, but you can’t see it; it is invisible to most antiviral programs.”

Apple Computer did create and send the program, Virus Rx, free to all bulletin board operators. Virus Rx allows the user to check files for most viral-type activities without knowing how a particular virus is trying to do its dirty work. Virus Rx can examine groups of files and check for things like hidden INITs in the system, resources embedded in files that wouldn’t normally contain resources. In McAfee’s scheme, Virus Rx would be classified as a Class Two program.

CONSIDER OTHER POSSIBILITIES

Viruses are by no means the only or even primary cause of a system crash or failure. Before you blame a virus for your ills, consider such causes as software bugs, disk controller failure, conflicts with memory-resident programs like Sidekick, incompatibility between software and hardware, or even human error.

John McAfee notes, “We do a lot of diagnostics that have nothing to do with viruses, but you have to do that in order to find the viruses.”

Vaccine Corporate by Foundation Ware is a program that goes far beyond simply checking for viruses. Vaccine Corporate’s creators realized that hard disk crashes and subsequent loss of data were the major concern of computer users, and that viruses were simply another means for this data loss to occur. The program is designed to detect changes or bugs in the software caused by viruses, bombs, power surges, static electricity, and cross-linking of files. Vaccine Corporate also prevents modification of approved software on the system and tracks who uses what on the system when.

Of course, the consumer pays a lot more for all these extra features ($195), but the program helps reduce the overall fear of disk crashes, a major problem in the microcomputer environment.
None of these antiviral approaches are foolproof. McAfee has found that even a total purge of a computer system is no guarantee against reinfection. He has reported that three out of four of the installations he visits suffer a relapse within a week, usually from disks missed on the first go-around or software carried in from the outside. New strains, called retroviruses, are especially designed to withstand hard disk purges.

Taking no chances, the military is practicing a unique approach to handling viruses; if a hard disk is infected by a virus, that hard disk is taken off line, reformatted, and never put back on line. It is never again trusted. According to an inside source, there are stacks of hard disks around military installations. Military personnel reformat them, then write nothing but zeros to the entire disk. Then they read them to make sure each disk contains only zeros, and then they reformat them again and delegate them to a storage room, never again to be used. Apparently the military has rooms full of reformatted hard disks, collecting dust. This is a rather extreme approach to handling the viral threat, but it does work.

There are some promising potential cures for viruses, discussed in our next chapter, "Viral Futures." But beyond the technology of viral cures, there are also the human, psychological factors, the reasons why people would motivate themselves to create and distribute viruses in the first place.

### THE PSYCHOLOGICAL CURE

Psychologist Dr. Robert Lambert, of Concordia University in Montreal, sees status for the hacker as a big motivation for viral creation, and a reduction in status as a cure for the problem.

"Think of computing twenty-five years ago. If you could program anything, if you could write a little twenty-five-line program in BASIC, you were hot stuff."
Nowadays, practically everybody can write a BASIC program. So merely being a programmer is no big deal anymore. Seventy-five years ago, it was a rare person who could drive a car. Fifty years ago, it was the rare bird who had a pilot's license.

“Reduction in status is the only thing that is going to cure the virus makers. Creating viruses must reach the stage where any two-bit hacker with a fifth-grade education, and who has computer experience is going to be able to create them.

“The human mind’s tendency to propagate according to the cookies one gets is a phenomenon of prodigious importance. If the mind stops getting kicks out of one thing, it will turn to something else.

“I suspect some time will pass before the human mind stops getting a kick from creating computer viruses. I predict the viral phenomenon will peak over a ten-year period. But for the most part, the viruses will eventually transmute themselves out of existence.”
Welcome to the Future and the Virus Activity Hotline

Where we keep you informed of the latest data on:

Heavy Hitters—System Crashers—Stealth Detection
Captured Players—Rookie Rampagers
Virulent Veterans—Power Punchers

TOP OF THE NEWS

IBM Loses Air Traffic Grid to Magick Mama
What Will She Do?
20,000 Human and Silicon Lives at Stake
Moment-by-Moment Coverage!
International Banks Bow to the Paralyzer
Claims to Be Ally of Robin Hood Virus

Medical Breakthrough
Silicon Virus Grows Computer Chip Inside Human Brain
Says It Feels Funny Feeling Feelings

SPRINTS

Latest Stats On
Magick Mama—Pakistani Brain MX # 3
OmniScores—zVIR2—Peace Machine
Venereal Veteran—Neutrino Man

The Daily Box Scores

The Viruses

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>zVIR2</td>
<td>Boot</td>
<td>UNIX</td>
<td>10</td>
<td>1,100</td>
<td>88</td>
<td>.880</td>
</tr>
<tr>
<td>Pakistani MX#3</td>
<td>Com</td>
<td>IBM PC</td>
<td>12</td>
<td>1,250</td>
<td>85</td>
<td>.850</td>
</tr>
<tr>
<td>Peace Machine</td>
<td>Boot</td>
<td>Mac</td>
<td>14</td>
<td>1,500</td>
<td>71</td>
<td>.710</td>
</tr>
<tr>
<td>Sin Scores</td>
<td>Boot</td>
<td>Mac</td>
<td>19</td>
<td>2,000</td>
<td>70</td>
<td>.700</td>
</tr>
<tr>
<td>Venereal Veteran</td>
<td>Com</td>
<td>IBM PC</td>
<td>23</td>
<td>4,500</td>
<td>65</td>
<td>.650</td>
</tr>
<tr>
<td>The Paralyzer</td>
<td>Com</td>
<td>UNIX</td>
<td>22</td>
<td>4,000</td>
<td>63</td>
<td>.630</td>
</tr>
<tr>
<td>Magick Mama</td>
<td>Boot</td>
<td>IBM PC</td>
<td>36</td>
<td>6,000</td>
<td>61</td>
<td>.610</td>
</tr>
</tbody>
</table>

The Antiviruses

<table>
<thead>
<tr>
<th>Heavy Hitters</th>
<th>Infector</th>
<th>Target</th>
<th>No. Kills/ Day</th>
<th>Percent Killed</th>
<th>Pct.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vaccine</td>
<td>Boot</td>
<td>IBM PC</td>
<td>4,000</td>
<td>80</td>
<td>.800</td>
</tr>
<tr>
<td>The Neutralizer</td>
<td>Com</td>
<td>Mac/PC</td>
<td>3,700</td>
<td>76</td>
<td>.760</td>
</tr>
<tr>
<td>Saviour</td>
<td>Boot</td>
<td>UNIX</td>
<td>3,500</td>
<td>73</td>
<td>.730</td>
</tr>
<tr>
<td>Kill Sin Scores</td>
<td>Com</td>
<td>Mac</td>
<td>3,300</td>
<td>69</td>
<td>.690</td>
</tr>
<tr>
<td>Peace Patrol</td>
<td>All</td>
<td>All</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
RELIGION

Prayer Virus Hits Automated Confessionals at Church of Heuristic Information Processing (C.H.I.P.)

Confessionals began dispensing instant absolution for 7 sins or more instead of less!

Is St. Silicon Really an Android?
Inquiring Cyborgs want to know.

-------PRESS ENTER FOR MORE--------

More indeed. The viral future is wide open, dependent on the minds of our species. Viruses can help us do anything we want, be it for the betterment or the downfall of us all. The choice is up to us. If, as Dr. Lambert suggests, destructive viral activity can be curtailed by reducing the status of viruses’ use for criminal purposes, then it is up to us individually and as a group to increase our viral literacy quotient so that any fifth-grader can design and build a virus.

This appears to be the natural direction of things, anyway. Computer literacy is increasing daily in this country, especially among our children. When given an opportunity to play with the machines, they learn computer languages like spoken languages. The oldest of these computer-literate kids are the hackers of today, mostly in their twenties. Used to a tremendous amount of freedom in the cyber-environments, they don’t fit into the more rigid structure created by the pre-computer-literate generation.

Increased computer literacy increases viral awareness which involves an understanding of security and defense, leading to increased chances of survival for those who know how to use and apply the new knowledge.

The brightest of our computer-literate kids know this path already. Thousands have played the game Core War
and know what it means to create and use battle programs. Dr. Cohen is teaching courses on what he knows about viruses. The Pakistani Brain virus is being used on college campuses as an example of how viruses work. John McAfee has distributed hundreds of sample viruses so that people can see and understand their function.

Victims of viruses will increase as long as there is ignorance of viruses. It is funny how victimhood and ignorance go hand in hand. But, as Dr. Cohen says, "That's life!" One can only hope to learn not to be a victim from being one. That often seems to be the lesson.

Viral politics are still in a primitive state. It is painfully obvious that the legal system cannot keep up with the hackers. Our law enforcement officers will catch only those viral designers who are blatantly obvious in their actions. And even then, it will take a computer-literate agent to prosecute.

The ascension to power by the baby-boom generation will be a result partly of increased computer literacy and awareness of a new world barely comprehended by the previous generation. The newly forming computer networks are linking us all, creating a planetwide family known as Homo sapiens sapiens.

As Bill Atkinson of Apple Computer says, "The computer networks are an externalization of the DNA." Individual selves are linking up via the computer networks, almost like the nervous system itself. Being part of this interconnected cyberworld grid is a far more powerful kind of awareness than has ever before existed for our species. It is a group mind of unprecedented proportions. With information on demand, it ultimately spells the end of ignorance as we have known it.

A definite moral responsibility comes with greater knowledge, especially knowledge that can affect the individual lives of millions of people. Our culture has still not awakened to the awareness that there is enough for all, choosing instead to act out of selfish individualism—look out for oneself and, maybe, one's family.

At some point, we must all make a choice as to who
and what we serve, be it ourselves, our species, our planet. I believe all three can be in alignment, but sometimes it requires a surrender of what we think is important.

The computer virus is a powerful phenomenon given to the masses of humanity. It will be an important factor in determining what institutions survive. The truly important structures we have created as a species will recreate themselves. The institutions that we don’t need will probably cease to exist.

I choose to use my newly acquired viral knowledge to improve my experience of being alive and to support the lives of those around me who are also dedicated to improving the quality of life experience. Being a communicator, I am also dedicated to improving the quality of communications among all sentient beings, a task that will keep me busy for some time to come.

I do not see computer viruses as the enemy. Instead, I see them as a challenge that will spur us to new heights of insight into the nature of who we are. The very idea of a virus is a door into the nature of existence, a key to living and understanding life more fully. The danger of a wrong move is death. The rewards of a right move just might be the elixir of life itself, eternal existence, should one so desire.

**MAKING THINGS BETTER**

Io I γασε ψου εσφερετινγη τηη ρω οων, ανδ ασκε δ φορ νοτηηγη ην ρετυρν, ωολδ ψου δο τηη σαμε φορ με ασ ι αολδ ψορ ψου?

Οη, τακε με φορ α ριδε, ανδ στριμ με οφ εσφερετινγη νυχλυδηη μψ βριδε, βυτ σπριτ ισ σομετηηγη τηη ιν ονε δεστροψο.
We were children once, playing with toys.
—Traffic

I predict that within the next ten years, certain species-survival beliefs like scarcity, (that there is not enough wealth to go around) will end. As the genuine perception grows that the entire species is our family, and that the earth is our home, there will be greater security and health for all. And the computer networks, as they become accessible to more people and become more usable, will be an integral part of our species' brain, providing practically unlimited intelligence, making whatever we want to happen, happen. And the computer viruses, like human ideas, will help spread the word, playing their part, successfully interlinking the networks. The word is actually three: going beyond survival. I believe it is our basic nature to make things better, not worse.

The future is what we choose to make it. With these thoughts in mind, I present you with some potentially effective solutions to the virus threat:

To make things better, in my opinion, we need to link the technology, the domains of the mind, with other realms of human experience, such as those of the heart. A successful future depends on the cross-pollination of these two very different domains of human experience. The new generation of computer literates are more normal kids than the older technogeneration, less nerdy, more aware of human dynamics and relationships. This is a good sign, a sign of a balance between the heart and mind. A sign of artists dropping their fears of math and science, a sign of the mind crew becoming aware of their bodies and the emotional mysteries of creation, a sign of going beyond survival.

Transcending issues of ego is paramount for going beyond survival. With the elimination of ignorance and the development of personal power, more of us will
realize that status seeking is a boring preoccupation and a waste of talent and time. We are all individuals with special talents and capabilities that are designed to mesh with everyone else's, creating collectively the body of humankind, an integrated species organism. The planet as a whole is an integrated organism, in which we are included, should we choose to look. If we choose to see ourselves as a disease on the surface of the planet, a planetary viral infection, then we will be treated as such.

How do you choose to perceive yourself in relation to the planetary organism?

Whatever your choice, your free will will be respected, and you will experience the consequences of your choice. This process is evolution in action, and we are all a part of it. Survival and going beyond survival are up to us, collectively and individually.

Evolving our computers to have an immune system is a definite emerging pattern of the near future. Our human immune system provides an excellent reference model for the design of future computer systems and has already been embraced by many of today's high-tech renaissance minds.

Designing systems to utilize viruses rather than simply battle them is a paradigm shift in thinking, embracing the phenomenon rather than running away from it. I predict this to be a viable path for our cyberfuture.

THE CYBERBEAST NETWORK:

A Modest Proposal

A computer network dedicated to the evolution of viruses, worms, and other cyberbeasts.

Purpose

1. To provide a network whereby people, who need to feed their creativity by researching viruses, can have an outlet for their experimentation and entertainment without having to misuse this creativity by breaking into and destroying other computer systems.
2. To evolve an understanding of these phenomena by all who are interested.

**Description**

An interactive, worldwide computer network isolated from all other computer networks and dedicated to cyberbeast activity. Essentially, the network will be a virtual world with its own rules. It will be a frontier, much like our own wild West of a hundred years ago. Within this frontier will be Cyber City, home base for bestiary research and development. Besides hosting academic forums on cyberbeasts and viral solutions, the virtual city will boast a Roman-like coliseum where beasts can be pitted against each other, a true testing ground for new creatures.

In this frontier environment, we will use our collective mind to learn all about these beasts, what they are and what they can do. Experimental territories will be set up, protected by whatever means its residents can imagine. Rewards of territory, recognition, and money will be given to those who can break through existing security systems and to those who evolve better defenses.

A colorful, real-time graphic user interface of Cyber City will make the town's activities interesting to less technical people. There must be a good graphic display showing progress of cyberbeast attacks while they are occurring. Statistics must be presented in real time as at a sporting event. There are the equivalent of generals, who are like football coaches. They manage and monitor actions and facilities like progress of the battle, the training of combatants, the bullpen, team strategies, and selling players to the highest bidder.

Bets may be placed on cyberbeasts in the coliseum so that money may be raised to support the network and reward the best players. The structure of the coliseum will be carefully defined so as to be in alignment with real-world problems.

New concepts and ideas will be the norm in Cyber City. Sci-fi network ideas like those espoused by William Gibson in *Neuromancer* can be tried, to see if they have real
value. Cyber City players might, for example, make use of personal defense tools:

**Cyber City Coliseum Defense Tools**

**Cyberbombs:** The right to destroy your own data. To be used only in case of emergency, they completely obliterate everything on your hard or floppy disk and/or launch themselves onto any connecting network/machine. Used for sacrificial moves.

**Ice:** A defense shield through which any information entering into your space is screened for abnormal characteristics. Some of the fancier models come with advanced retaliatory capabilities (such as trapping the intruder, secretly identifying the location of its home-base computer, dialing it up unsuspectingly, and dumping a cyberbomb into its home system, obliterating its memory, then suddenly dropping the connection).

**Ice Breakers:** Programs that penetrate defense shields and perform the deeds of their sender. Typical activities include password searching, inserting cyberbombs, and collecting useful data (such as bank account numbers, industrial secrets, or legal records).

**Memory Defense Systems:** Short-term invader-alert tools to use when under attack.

**Memory Blockers:** Devices that secure files and programs from penetration during attack (like the wagon trains forming a circle when the pioneers are seriously threatened, and about as effective).

**Self-Destruct RAM Dumper:** Program that, upon command, dumps information in random access memory (RAM), causing an immediate reboot of your system (comparable to blowing the ballast on a submarine).

**Device Sweeper:** Program that looks for abnormal program and system behavior, obliterating and/or isolating it.
Cyber City might have a resident’s manual that contains helpful suggestions like the following:

Every PC owner has a right to protect him- or herself in the event of an attack by hostile forces. Each should be well equipped and trained in the uses of cyberbeast warfare, both defensive and offensive.

The best way to win a war is to win before you get started, before your opponent even begins to get ready. Merciless, but true... in the frontier worlds of cyberspace. No white picket fences here.

Descriptions of Cyber City can be couched in terms of the freedoms of the wild West:

Cyber City—famed for its cyberbeast gambling arenas, munitions, and entry into cyberspace politics, where residents ‘discuss’ relations between the various centers of cyberspace activity and its satellite worlds (a way of linking certain people and moving energy around the planet). A sign entering Cyber City reads, ‘Welcome to Cyber City—A Good Place to Think From.’

Cyber City is a radical solution to the problem of computer viruses, but then again, the viruses are a radical problem. Sometimes it takes unusual methods to achieve unusual results.
As of May 1989, dozens of viral software "fixes" have appeared on the market, giving us all the feeling that something is being done about the issue. Viruses are no longer headline news, replaced with articles reflecting the computer industry's more immediate interests—higher-resolution screens, faster memory and processing speeds, more color, networking, oriental clones, and friendlier user interfaces. I predict this lull in viral activity is a temporary respite to be followed within the next few months by a second wave of action.

The first wave of viral attacks was created by computer hackers seeing what they could do with the concept. Now that the concept has been field-tested, the mainstream computer users will "have at it." These mainstream users have different motives than the hackers. Rather than focusing on experimentation, they will probably use viruses for increasing personal status, recognition, and cash flow.

One new virus (not yet widely known) could give you Triple A rating on your credit cards, regardless of your
past credit history. Another is designed to infiltrate the Automatic Teller Machine bank systems. I have also heard rumors of viral problems in the IRS, where massive amounts of data on personal income taxes may have been lost.

These stories have not yet been reported in the mainstream media. In the meantime, more than one of the viral designers has reported to me that they believe they are under surveillance and are therefore keeping a low profile.

The truth is that a deadly network virus, already conceived by Dr. Fred Cohen, could easily erase every file from local area office networks to planetary-wide computer grids, such as the international banking system. This virus could be designed by a reasonably competent individual in less than 100 hours—and there is very little we can do to protect our computer systems.

At present, civilian computers are most at risk because, until recently, there has not been the funding nor the urgency to develop effective countermeasures to viruses. The Internet Worm has served as a very effective warning of the potential danger, causing the computer industry to take notice. Conferences are hurriedly being held on viruses by security experts around the world, but it will take at least a couple of years to develop any really good protection from computer viruses.

There is no panacea, no sure-fire cure—there may never be one. The best guidelines for immediate protection come from Delbert Jones, founder of the National LAN Laboratory in Reston, Virginia. Endorsed by over 70 computer product manufacturers and vendor organizations, his guidelines are as follows:

1. All software should be purchased from known, reputable sources.

2. All purchased software should be in its original shrink wrap or sealed diskette containers when received.
3. Backup copies of all original software should be made as soon as the software package is opened. Backup copies should be stored off-site.

4. Once purchased, all software should be reviewed carefully by a system manager before it is installed on a distributed system.

5. New software should be quarantined on an isolated computer. This testing will greatly reduce the risk of system virus contamination.

6. A backup copy of all system software and data should be made at least once a month, with the backup stored for at least one year before reuse. This will allow restoration of a system that has been contaminated by a "time-released" virus. A plan that includes "grandfathered" rotation of backup copies will reduce risk even further.

7. System administrators should restrict access to system programs and data on a "need-to-use" basis. This isolates problems, protects critical applications, and facilitates problem diagnosis.

8. All programs on a system should be checked regularly for size changes. Any size deviations could be evidence of tampering or virus infiltration.

9. Many "shareware" and "freeware" programs are invaluable applications. However, these programs are the prime entry point for system viruses. Skeptical review of such programs is prudent. Also, extended preliminary quarantine is essential before these programs are introduced on a distributed system.

10. Any software which exhibits symptoms of possible virus contamination should be removed from general use immediately. System managers should develop plans for quick removal from service of all copies of a suspect program, and for immediate backup of all related data. These plans should be made known to all users, and tested and reviewed periodically.
Many kinds of antiviral software will become available over the next few years. Some of it will work and the others will be junk. For example, Michael Synergy in the San Francisco Bay area is pioneering a new approach using a combination of hardware and software to prevent viral infection. In the end, though, one of the best forms of protection is to keep up on antiviral product reviews in reputable trade magazines relevant to your type of computer.

Besides new antiviral software, there will be lots of "computer security" consultants appearing out of the woodwork. Some will be quite talented, many will be con-artists. Industry analyst Wes Thomas warns that there will be a "viral protection racket" forming in this country:

When there is money to be had, there will always be peddlers with something to sell, who will try to convince you that their method is best. With respect to viruses, they may claim to be "security experts," but many of them will be Watergate types, CIA renegades. They will seek out vulnerable companies, meet with the CEOs, and say, "We will protect you from viruses, but it is going to cost you because your company has enemies and they will be out to get you. We know who they are and we can protect you, but it's going to take money."

First the target company will say "no." Then, inevitably, within a few weeks they get hit by a virus. The protection boys will come back and say, "I hear you got hit. We warned you about that!" There is no easy way to prove that they were responsible for the hit. So, be very careful in dealing with security consultants. Check them out first before you hire them.

As far as the government is concerned, its reaction will probably be to try to make viruses illegal, except to agencies of its choosing. There is already an antiviral bill before Congress (HR 55). This approach is unworkable
because there are over 30 million computers out there on which viruses can be created, and the only way such a law could be enforced would be to make all personal computers illegal.

Now or in the future, your best defense is an awareness of your vulnerability and extreme vigilance when allowing access to your system.
APPENDIX A
ANALYSIS OF THE INTERNET WORM

Is it a virus? Is it a Trojan? No, it's a Networm!

Several analyses of the Internet Worm appeared on the computer networks within the first few weeks following its attack.

One of the first was from Bob Page at the University of Lowell on November 7, 1988. In a simple five-page summary distributed on the Internet, he outlined the three basic methods of worm attack, the anatomy of the worm itself, and how the worm was stopped.

One of the most thorough analyses was Purdue Technical Report CSD-TR-823, issued by Eugene H. Spafford on November 29, 1988. Spafford gives us a detailed description of the worm's components, data, and functions, basing his analysis on two completely independent reverse-compilations of the worm.

The complete original report by Eugene H. Spafford, "The Internet Worm Program: An Analysis," can be found in the Association for Computing Machinery's Computer Communication Review 19(1), Jan. 1989. Portions of the report reproduced here are © 1988 by Eugene H. Spafford. All rights reserved. Used by permission.
During the attack, Spafford set up an emergency information system, providing many people with up-to-date reports about the intruder. He was even able to catch one of the worms as it tried to penetrate his computer.

The National Computer Security Center took notice of Spafford's actions and sent him a special invitation to a hastily convened worm workshop in Baltimore on November 8, 1988. Who attended, why they were invited, and the topics discussed have not yet been made public. But this much we know: those who attended decided not to distribute copies of the worm code to the general public. It was felt that the worm exploited too many little-known programming techniques, and making it available would only provide other attackers a framework to build another such program.

Spafford found this security stance somewhat amusing, saying, "As of November 27, I am aware of at least five versions of the decompiled code, and because of the widespread distribution of the binary, I am sure there are at least ten times that many versions already completed or in progress—*the required skills and tools are too readily available within the community to believe that only a few groups have the capability to reconstruct the source code.* Even if none of these versions is published in any formal way, hundreds of individuals will have had access to a copy before the end of the year. *Historically, trying to ensure security of software through secrecy has proven to be ineffective in the long term.*"

Secrecy for the sake of security simply doesn't work here. In this case, the more people who know about the worm—especially the system administrators, programmers, and managers—the more secure the network will be. The main interest is not in duplicating the worm but in plugging the potential wormholes existing within the network.

So, with openness in mind, we delve a little more deeply into the nature of the Internet Worm.

By definition, the Internet program is a worm and not a virus because a worm can exist and breed by itself,
while a virus cannot run independently, but requires a "host" as a home.

Exactly how the worm penetrated Internet security is of great interest. The basic object of the worm is to get a shell on another machine so it can reproduce further. It used three methods of system penetration: the Fingerd attack, the Sendmail attack, and the password attack.

THE FINGERD ATTACK

Fingerd is a utility program similar in function to a telephone book or information operator. It is used for identifying the names, phone numbers, and addresses of system users, and when they were last on the network. The Fingerd program is designed to run as a background or daemon process, a service on your electronic desk that can be collecting data for you while you are doing other things on the system.

The worm took advantage of Fingerd by overrunning the daemon's input buffer, by bombarding it with too much information too fast. This caused the daemon to malfunction, allowing the worm to slip past the "electronic information operator" and into the local system for which the operator was responsible.

The worm overran the buffer by exploiting the gets call, which takes input to a buffer without doing any bounds checking. According to Page, this attack was where most of the worm's success occurred:

When Fingerd is connected, it reads its arguments from a pipe, but doesn't limit how much it reads. If it reads more than the internal 512-byte buffer allowed, it writes past the end of its stack. After the stack is a command to be executed ("/usr/ucb/finger") that actually does the work. On a VAX, the worm knew how much further from the stack it had to clobber to get to this command, which it replaced with the command "'/bin/sh" (the Bourne shell). So instead of the Fingerd command being
executed, a shell was started with no arguments. Since this is run in the context of the finger daemon, stdin and stdout are connected to the network socket, and all the files were sucked over just like the shell that Sendmail provided.

Spafford points out that the standard C library has a number of routines that read input without checking for bounds on the buffer involved:

The family of routines \texttt{scanf/fscanf/sscanf} may also overrun buffers when decoding input unless the user explicitly specifies limits on the number of characters to be converted. Incautious use of the \texttt{sprintf} routine can overrun buffers. Use of the \texttt{strcat/strcpy} calls instead of the \texttt{strncat/strncpy} routines may also overflow their buffers.

What's interesting here is that experienced C programmers are aware of these bugs, yet they continue to use the routines. Further, these routines are included in the forthcoming ANSI language standard for C. One solution would be to formulate a set of replacement calls with bounds on the buffers, a project of Spafford's.

\textbf{THE SENDMAIL ATTACK}

Sendmail is a complex program designed to route electronic mail throughout the Internet. It has many features and modes, but it is of most interest to us when it is working as a daemon background process. In this mode, the program is like a sentry at his post interrogating people who wish entry into the compound he is guarding. In this case, the Sendmail daemon is at his post (TCP port #25) "listening" for incoming Internet mail to his base computer system. He listens for SMTP (Simple Mail Transfer Protocol), a standard Internet language. When he receives a request for access, the daemon en-
ters into a dialog with the remote mailer to determine sender, recipient, delivery instructions, and message contents.

The worm slipped past the sentry by using false ID, the equivalent of a technician's pass. When queried by the daemon, the worm responded with a DEBUG command. This put the daemon "at attention" ready to receive orders. The worm then issued a special set of commands instead of the usual user address and message routine.

Normally, this is not allowed, but the debug code allows system testers to verify that mail is arriving at a particular site without the need to activate the address routines. The debug option of Sendmail is often used because of the complexity of configuring the mailer for local conditions, and many vendors and site administrators leave the debug option compiled in.

Once the worm invoked DEBUG mode, it sent a "rcpt to" that requested its data be piped through a shell. That data, a shell script (first-stage bootstrap), created a temporary second-stage bootstrap file called x$$,11,c (where "$" is the current process ID). This file was short, a forty-line C program. Page describes the next phase as follows:

The first-stage bootstrap compiles this program with the local cc and executes it with arguments giving the Internet hostid/socket/password of where it just came from. The second-stage bootstrap (the compiled C program) sucks over two object files, x$$,vax.0 and x$$,sun3.0 from the attacking host. It has an array for 20 file names (presumably for 20 different machine types), but only two (VAX and Sun) were compiled into this code. It then figures out whether it's running under BSD or SunOS and links the appropriate file against the C library to produce an executable program called /usr/tmp/sh—so it looks like the Bourne shell to anyone who looked there.
THE PASSWORD ATTACK

In the password attack, the worm attempted to discover user passwords. It was successful because an encrypted version of all system user passwords was available in a publicly readable file. This public file allowed the worm to encrypt lists of possible passwords and then compare them against the actual passwords without passing through any system function.

To further figure out people's passwords, the worm tried guessing. It used combinations of the user name—the last, first, last plus first, nicknames—and an onboard list of special popular passwords, which is reproduced at the end of this appendix.

If these techniques failed, the worm opened the system dictionary (/usr/dict/words) and tried every word. Its password-cracking procedure was pretty successful, as most people do not choose their passwords very carefully.

Once the worm got into someone's account, it attempted to break into other machines where that user had accounts. To do this, it looked for a .rhosts file and did an "rsh" and/or "rexec" to another host. It then sucked over the necessary files into /usr/tmp and ran /usr/tmp/sh to start the process all over again.

Using these three methods of attack, the worm was able to penetrate over six thousand computer systems in less than twenty-four hours.

A TOUR OF THE WORM

The worm has two major segments, each further divided into several subsegments. The major segments are called global data structures and routines.

The global data structures consist of several lists and arrays used for collecting, storing, and disseminating information at the appropriate times. The Host list, for example, stores up to twelve host names and aliases. The Gateway list uses the network netstat command to col-
lect addresses used to infect directly connected networks. The 432-name Password list was used for cracking user passwords. Embedded masked strings were used for concealment purposes and to complicate analysis of the worm.

The worm routines are what make the worm do things. Spafford grouped them by function, listing them as follows:

- **Setup and utility routines**—main, doit, crypt, h_addaddr, h_addname, h_addr2host, h_clean, h_name2host, if_init, loadobject, makemagic, netmaskfor, permute, rt_init, supports_rsh, and supports_telnet.

- **Network and password attacks**—attack_network, attack_user, crack_0, crack_1, crack_2, crack_3, cracksome, ha, hg, hi, hl, hul, infect, scan_gateways, sendworm, try_fingerd, try_password, try_rsh, try_sendmail, and waithit.

- **Camouflage**—checkother, other_sleep, send_message, and xorbuf.

Spafford’s report discusses each routine in some detail. For tracing the basic functional flow of the worm, the *main* and *doit* descriptions are quoted:

5.2.1 main

This was where the program started. The first thing it did was change its argument vector to make it look like it was the shell running. Next, it set its resource limits so a failure would not drop a core file. Then it loaded all the files named on the command line into the object structure in memory using calls to loadobject. If the ll.c file was not one of the objects loaded, the worm would immediately call exit.

Next, the code unlinked all the object files, the file named sh (the worm itself), and the file /tmp/.dumb
(apparently a remnant of some earlier version of the program, possibly used as a restraint or log during testing—the file is not otherwise referenced). The program then finished zeroing out the argument vector.

Next, the code would call `if_init`. If no interfaces were discovered by that routine, the program would call `exit`. The program would then get its current process group. If the process group was the same as its parent process ID (passed on the command line), it would reset its process group and send a KILL signal to its parent.

Last of all, the routine `doit` was invoked.

5.2.2. `doit`

This was the main worm code. First, a variable was set to the current time with a call to `time`, and the random number generator was initialized with the return value.

Next, the routines `hg` and `hl` were invoked to infect some hosts. If one or both of these failed to infect any hosts, the routine `ha` was invoked.

Next, the routine `checkother` was called to see if other worms were on this host. The routine `send_message` was also called to cast suspicion on the folks at Berkeley. As if some of them aren't suspicious enough!

The code then entered an infinite loop:

A call would be made to `cracksome` followed by a call to `other_sleep` with a parameter of 30. Then `cracksome` would be called again. At this point, the process would `fork` itself, and the parent would `exit`, leaving the child to continue.

Next, the routines `hg`, `ha`, and `hi` would all be called to infect other hosts. If any one (or combination) of these routines failed to infect a new host, the routine `hl` would be called to infect a local host. Thus, the code was aggressive about always infecting at least one host each pass
through this loop. The logic here was faulty, however, because if all known gateway hosts were infected, or a bad set of host numbers were tried in ha, this code would call hl every time through the loop. Such behavior was one of the reasons hosts became overloaded with worm processes: every pass through the loop, each worm would likely be forced to infect another local host. Considering that multiple worms could run on a host for some time before one would exit, this could lead to an exponential growth of worms in a LAN environment.

Next, the routine other_sleep was called with a timeout of 120. A check was then made to see if the worm had run for more than 12 hours. If so, a call was made to h_clean.

Finally, a check was made of the pleasequit and nextw variables (set in other_sleep or checkother, and crack_2, respectively). If pleasequit was non-zero, and nextw was greater than 10, the worm would exit.

With respect to the intent of the Internet Worm, Page says, "Its role in life is to reproduce—nothing more."

Spafford elaborates, "I can state with some certainty that if there was only one version of the worm program, then it was benign in intent. It did not write to the file system except when transferring itself into a target system. It also did not transmit any information from infected systems to any site, other than copies of the worm program itself."

**THE WORM'S STYLE AND STRUCTURE**

What may surprise some readers is that the quality of the worm code is mediocre, and might even be considered poor. There are places, for instance, where calls are made to functions with either too many or too few arguments. Many routines have local variables that are never
used. In at least one location, a struct is passed as an argument rather than the address of the struct. There are also dead code routines that are never referenced, and code that cannot be executed because of conditions that are never met (possibly bugs).

At many places in the code, there are calls on system routines with return codes that are never checked for success. Calls are made to the system heap routine, malloc, and the result is immediately used without any check. Although the program was configured not to leave a core file or other evidence if a fatal failure occurred, the lack of simple checks on the return codes is indicative of sloppiness, also suggesting that the code was written and run with minimal or no testing. Says Spafford:

The structures chosen for some of the internal data are also revealing. Everything was represented as linked lists of structures. All searches were done as linear passes through the appropriate list. Some of these lists could get quite long, and doubtless that considerable CPU time was spent by the worm just maintaining and searching these lists. A little extra code to implement hash buckets or some form of sorted lists would have added little overhead to the program, yet made it much more efficient (and thus quicker to infect other hosts and less obvious to system watchers). Linear lists may be easy to code, but any experienced programmer or advanced CS student should be able to implement a hash table or lists of hash buckets with little difficulty.

Some effort was duplicated in spots. An example of this was in the code that tried to break passwords. Even if the password to an account had been found in an earlier stage of execution, the worm would encrypt every word in the dictionary and attempt a match against it. Similar redundancy can be found in the code to construct the list of hosts to infect.

There are locations in the code where it appears that
the author(s) meant to execute a particular function but used the wrong invocations. The use of the UDP send on a TCP socket is one glaring example. Another example is at the beginning of the program where the code sends a KILL signal to its parent process. The surrounding code gives strong indication that the user actually meant to do a killpg instead but used the wrong call.

There was one section of the code that stood out from the rest. It was particularly well thought out, involving the crypt routines used to check passwords. As has been noted, this code is nine times faster than the standard Berkeley crypt function. Many interesting modifications were made to the algorithm, and the routines do not appear to have been written by the same author as the rest of the code. Additionally, the routines involved have some support for both encryption and decryption—even though only encryption was needed for the worm. Perhaps this routine was written by someone other than the author(s) of the program, and included with the code. It would be interesting to discover where this code originated and how it came to be in the worm program.

The program could have been much more virulent had the author(s) been more experienced or less rushed in his, her, or their coding. It's likely that this code had been developed over a long time, indicating that the author(s) was sloppy or careless (or both) and perhaps that the release of the worm was premature.

The same attack used on the finger daemon could have been extended to the Sun version of the program but was not. The only explanations that come to mind why this was not done are that the author lacked the motivation, the ability, the time, or the resources to develop a version for the Sun.

Spafford reports that at a recent meeting, Professor Rich Rashid of Carnegie-Mellon University was heard to claim that Robert T. Morris, Jr., the alleged author of the worm, had revealed the Fingerd bug to system administrative staff at CMU well over a year ago. Assuming this report is correct and the worm author is indeed Mr.
Morris, it is obvious that there was sufficient time to construct a Sun version of the code.

In fact, Spafford asked three Purdue graduate students to develop a Sun 3 version of the attack, and they did so in under three hours. The worm author certainly must have had access to Suns, or else he would not have been able to provide Sun binaries to accompany the operational worm. Nor should motivation be a factor considering everything else present in the program.

Spafford wrote, “With time and resources available, the only reason I cannot immediately rule out is that he lacked the knowledge of how to implement a Sun version of the attack. This seems unlikely, but given the inconsistent nature of the rest of the code, it is certainly a possibility. However, if this is the case, it raises a new question: was the author of the worm the original author of the VAX Fingerd attack?”

Propagation and Load

Perhaps the most obvious shortcoming of the worm code is the lack of understanding about propagation and load. The worm was spotted so quickly and caused so much disruption because it replicated itself exponentially on some networks, and because each worm carried no history with it.

Many approaches could have been taken by the author(s) to slow the growth of the worm or prevent reinfection. Their absence from the worm program is telling. Either the author(s) did not have any understanding of how the program would propagate, or else he, she, or they did not care; the existence in the worm code of mechanisms to limit growth tends to indicate that it was a lack of understanding on the part of the creator(s) rather than an indifference to the issues.

Some of the algorithms used by the worm were reasonably clever. One in particular is interesting to note: when trying passwords from the built-in list or when trying to break into connected hosts, the worm would randomize
the list of candidates for trial. Thus, if more than one worm were present on the local machine, they would be more likely to try candidates in a different order, thus maximizing their coverage.

This implies that the author(s) was not overly concerned with the presence of multiple worms on the same machine. More to the point, multiple worms were allowed for a while in an effort to maximize the spread of the infection. This also supports the contention that the author did not understand the propagation or load effects of the worm.

The design of the vector program, the "thinning" protocol, and the use of the internal state machine were all clever and not at all obvious.

The overall structure of the program, especially the code associated with IP addresses, indicates considerable knowledge of networking and the routines available to support it. The knowledge evidenced by that code would indicate extensive experience with networking facilities.

Overall, we could conclude that the author was not a careful programmer—the code errors were largely a result of ignorance or inexperience.

Camouflage

Great care was taken to prevent the worm program from being stopped. This can be seen by the caution with which new files were introduced into a machine, including the use of random challenges. Every string compiled into the worm was encrypted to prevent simple examination. Files associated with the worm were carefully deleted from disk at the earliest opportunity, and the corresponding contents were encrypted in memory when loaded. It was evidenced by the continual forking of the process and the (faulty) check for other instances of the worm on the local host.

The code also took precautions against providing copies of itself to anyone seeking to stop the worm. It sets its
resource limits so it cannot dump a core file, and it keeps internal data encrypted until used.

Fortunately, there are many methods of obtaining core files and data images, and researchers were able to obtain all the information they needed to disassemble and reverse-engineer the code. There is no doubt, however, that the author(s) of the worm intended to make such a task as difficult as possible.

One question worthy of asking is why more machines weren't targeted for attack. Considering that the author(s) knew how to break into arbitrary UNIX machines, it seems odd that he, she, or they did not attempt to compile foreign architecture programs for inclusion in the worm code.

**SUMMARY AND CONCLUSIONS**

The creator(s) of the worm program may have been a moderately experienced UNIX programmer, but by no means the "UNIX wizard" many have been claiming.

The code employs a few clever techniques and tricks, but they may not be the original work of the worm author. The code is sloppy and reflects inexperience. Whoever put this program together appears to lack fundamental insight into some algorithms, data structures, and network propagation, but at the same time has some very sophisticated knowledge of network features and facilities.

The worm code does not appear to have been tested, and it is likely that it was prematurely released. The presence of so much dead and duplicated code coupled with the size of some data structures (such as the twenty-slot object code array) suggests that the program was intended to be more comprehensive.

It is clear from the code that the worm was deliberately designed to do two things: infect as many machines as possible and be difficult to track and stop.

The effects of this worm were largely benign, and it was easily stopped. Had the code been tested and devel-
oped further by someone more experienced, or had it been coupled with something destructive, the toll would have been much higher.

The major sources of information for many of the sites affected seems to have been Usenet news groups and a mailing list Spafford put together when the worm was first discovered. Although helpful, these methods did not ensure timely, widespread dissemination of useful information—especially since they depended on the Internet to work!

Rich Adams of the Center for Seismic Studies has commented that we may someday hear that the worm was loosed to impress Jodie Foster. Until more data are available, his is as valid a speculation as any other, and it raises the issue that not everyone with access to computers is rational and sane. Future attacks may reflect this fact.

The Internet Worm has shown us that we are all affected by events in our shared environment, and we need to develop better information methods outside the network before the next crisis begins. The next crisis is not a question of if. Now it is a question of when and where.

### Passwords Tested by the Internet Worm

<p>| aaa          | arrow | berliner | charles | dave | elizabeth | foolproof | football | foresight | engine | format | forsythe | fourier | friend | fun    | fungible | Gabriel | gardner | garfield | felicia | gauss | george | gertrude | glacier | gnu   | golfer | gorgeous | gorges | gosling |
|-------------|-------|----------|---------|------|-----------|-----------|----------|-----------|--------|--------|----------|--------|--------|--------|----------|---------|--------|--------|----------|--------|-------|--------|----------|--------|-------|--------|----------|--------|-------|</p>
<table>
<thead>
<tr>
<th>gouge</th>
<th>Graham</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gryphon</td>
<td>guest</td>
</tr>
<tr>
<td>handily</td>
<td>happening</td>
</tr>
<tr>
<td>harold</td>
<td>harry</td>
</tr>
<tr>
<td>Hebrides</td>
<td>Heinlein</td>
</tr>
<tr>
<td>helo</td>
<td>help</td>
</tr>
<tr>
<td>Herbert</td>
<td>Hiawatha</td>
</tr>
<tr>
<td>hibernia</td>
<td>Honey</td>
</tr>
<tr>
<td>Horse</td>
<td>Horse</td>
</tr>
<tr>
<td>Hutchins</td>
<td>Imperial</td>
</tr>
<tr>
<td>Imbroglio</td>
<td>Inna</td>
</tr>
<tr>
<td>Innocuous</td>
<td>Irishman</td>
</tr>
<tr>
<td>Isis</td>
<td>Japan</td>
</tr>
<tr>
<td>Jessica</td>
<td>jester</td>
</tr>
<tr>
<td>Jixian</td>
<td>Johnny</td>
</tr>
</tbody>
</table>
Amiga virus: A boot infector that prints the following message after infection has occurred:

Something wonderful has happened.
Your Amiga is alive!!!
and even better
Some of your disks are infected by a VIRUS
Another masterpiece of the Mega-Mighty SCA

Antivirus software virus: Conceived by Chuck Weinstock in a message on Usenet on February 9, 1988, a virus embedded in the software and used to detect viruses. It therefore goes undetected itself.

ARPAnet data virus: A virus that, in an early accidental mutation on October 27, 1980, caused multiple "status" messages to begin to appear on ARPAnet. Usually status messages are broadcast from each node of the network to relay their readiness to handle new data. Each of these nodes then propagates copies of incoming status messages to other nodes in an ongoing determination of the optimal path for the electronic traffic. Status messages are supposed to be trashed immediately afterward, but here a mutated node near Los Angeles caused a "garbage collector" malfunction in the receiving nodes. The messages were never thrown out, causing serious constipation and saturating the nodes. This constipation spread throughout the network, like a form of cancer, and ground it to a halt. It was three days before the network was functional again.
Atari ST/Mega virus: A virus that copies itself onto every non-write-protected disk used. After infecting a disk, it replicates and keeps count of the number of times the disk is used. It then writes random data across the disk's root (central) directory and file allocation tables (the computer's index of where data is stored), making the disk unusable. It then removes itself from the damaged disk. Though the virus doesn't affect Atari hard disks, it can survive a reset (resetting the machine without turning it off). It was first dissected on March 22, 1988, by Martin Minow and posted on Usenet on March 26, 1988.

Bell Labs virus: A virus that embedded a compiler program with a hidden trap door each time a new version of the operating system was created. The secret trap door altered the system so that, in addition to normal users' passwords, it would recognize a magic password known only to one person. Undetectable via normal means, the virus never escaped the domain of Bell Labs.

Brain virus: Originating in Pakistan, a boot infector that contains a message reading, "Welcome to the dungeon . . . Beware of the VIRUS. Contact us for vaccination." Originally designed for software-protection purposes, it is most famous for infecting the computer system of the Providence Journal-Bulletin. (See full Brain story in Chapter 3.)

Compressor virus: Created by Dr. Fred Cohen, a virus that is actually useful. It compresses data, so that it can be stored in a smaller space, asking the permission of the user each time before it acts.

Creeper: Created in 1970 and one of the earliest viruses on record, a virus that crawled through ARPAnet—a nationwide Pentagon-funded network linking university, military, and corporate computers—and sprang up on computer terminals with the message, "I'm the Creeper, catch me if you can!" It was created by Bob Thomas, of the Bulletin Board Network. A later version of Creeper was designed by Ray Tomlinson and moved through the net, replicating itself at times. See "Legends from the Cyberpast," in Chapter 2.

Elk Cloner virus: An Apple II virus that inserts itself into the DOS operating system, hooking into the RUN, LOAD, BLOAD, and CATALOG commands. It makes the commands check the accessed program disk and infect it, printing a poem:

The Program with a personality
It will get on all your disks
It will infiltrate your chips
Yes, it's cloner!
It will stick to you like glue
It will modify RAM too
Send in the cloner!
Appendix B: A Software Bestiary

Evolutionary virus: Designed by Dr. Fred Cohen at the University of Cincinnati, a virus that mutates its code with each successive replication, making it more difficult to track.

Finger virus: A tracking virus, designed to find a specific person's electronic mail address and send the data back to its creator. Imagined by Fred Hapgood at the First Artificial Life conference in September 1987.

Good virus: Written in West Germany, a virus that won't let "unknown" programs run on one's machine. If the programs to be run aren't already infected with this virus, they won't be allowed to run at all.

Immortal virus: This virus exists in some cacheliike memory on a serial or parallel port (what connects a printer to a computer), so it would survive a warm boot, even after a devirusing. Not as yet in existence, it was imagined by Paul Hoffman on March 13, 1988, in the Macintosh Conference on The Well.

Invisibility virus: A virus that attaches itself to the Macintosh antivirus program Interferon, appearing invisible to the system and Interferon. Discovered in the Bay Area in the summer of 1988 by an Apple employee, it is not known to have spread elsewhere as yet.

Israeli Time Bomb virus (Jerusalem virus): First sighted at Hebrew University, it was designed to destroy files on May 13, 1988, the fortieth anniversary of Israel as an independent nation. Fortunately, it had an error in its code, causing it to reinfect the same programs over and over again, sometimes infecting the same program as many as 400 times. This made it easy to discover, and it was disarmed before the big birthday party.

King virus: A virus that kills other viruses and replaces them with itself. Imagined by Andrew Beals on March 16, 1988, on the computer network The Well.

King II virus: A virus that kills other viruses and then feeds on them, getting stronger each time. Imagined by Michael Zentner on March 16, 1988, in the Macintosh Conference on The Well.

Lehigh virus: A COMMAND.COM infector, it destroys floppy and hard disk data by writing zeros to the first thirty-two sectors of a disk, making the data irrecoverable. It waits until it has been copied four times before it wipes out the data on the disk on which it resides. See Chapter 3 for the complete story.

Magik virus: An IBM PC infector, a virus that leaves the following message on the CRT:

****** HELLO FRIEND ******
WE ARE SORRY TO INFORM YOU THAT YOUR
SYSTEM IS VERY ILL  
AND WILL NOT SURVIVE 
MUCH LONGER 
YOU UNFORTUNATELY HAVE 
-THE BRAZILIAN BUG- 
FOR WHICH THERE IS NO CURE 
WE HOPE THAT THIS WILL NOT BE AN INCONVENIENCE TO YOU 
LOVE, 
THE MAGIK MUSICIANS 

After delivering this "friendly" message, the computer begins to erase everything on the hard disk or floppies. A definitely malicious act, by any definition. Technically, it is a very fast infector and reinfector. And after reaching activation stage, it does a low-level format of the drive. Its saving grace is that it displays the message before it destroys your data, giving you a little time to think about it. The message is not displayed, however, until your system is well infected (kind of like finding out you have cancer with only six minutes to live). Or as John McAfee (its collector) notes, "This virus appears after it has done its damage and there is nothing more you can do about it."

Metavirus: An imaginary paranoia virus, created with the following words:

WARNING! A serious virus is on the loose. It was hidden in the program called 1988 TAXFORM that was on this bulletin board last year. . . . By now, it is possible that your system is infected even if you didn't download this program, since you could easily have been infected indirectly. The only safe way to protect yourself against this virus is to print all your files onto paper, erase all the disks on your system, buy fresh software disks from the manufacturer, and type in all your data again. But FIRST! send this message to everyone you know, so that they will also protect themselves.

This message took Jeffrey Mogul two minutes to produce, and he didn't have to write any code! First posted on Usenet on February 9, 1988.

No-Name virus: A virus that is almost impossible to detect because, for each disk, it scans for any program file and appends itself to the text segment in some way. Rumored to exist by Martin Minow on Usenet, March 26, 1988.

nVIR: Written in West Germany and posted to a hypercard stack on the CompuServe network, it is a virus disguised as a Mac resource that inserts itself into a system trap handler (where the Mac catches errors that could cause system crashes). It destroys hard disks and has been known to say, "Don't Panic!" upon
bootup. Several mutations of this creature are in circulation at this time.

Peace virus (also known as the Brandow virus): Supposedly a simple message of peace, designed to pop up on Macintosh screens on March 2, 1988, the anniversary of the introduction of the Apple Macintosh SE and Macintosh II. The virus infects the SYSTEM file, but doesn't directly affect applications. Benign by design, the virus nonetheless had side effects that played havoc with users' SYSTEM folders, resulting in thousands of hours of lost work. It was the first virus to infect a commercially available computer product. See Chapter 3 for further details.

Pervading Animal: An early virus that was attached to a Univac 1108 game program called Animal. While the game was being played, Pervading Animal copied itself into every write-enabled program file available.

Rabbit: First sighted in 1974 by Bill Kennedy, a virus that replicated itself inside a mainframe computer system and then continuously tossed copies back into the input jobstream (where programs start). This greatly slowed the communication between the input jobstream and its console (teletype where the system operator sees what is going on), which made Rabbit harder to kill, the longer it ran.

Reaper: The counterforce to Creeper. This virus also jumped through computer networks, its purpose to detect and "kill" Creepers.

Scores virus: A major Mac virus. It has infiltrated several government agencies, Apple sales offices, and MacWorld and Macintosh Today. The virus causes printing problems, disk crashes, application crashes on launch, and damaged EXCEL files. See Chapter 3 for the detailed story.


Target virus: A virus that targets a specific program or individual by systematically altering data or performing other subtle changes.

Trojan horses: A relative of viruses, bits of code that also slip into other programs. They do not replicate like the viruses, but simply stay in one place and perform their destructive acts from there. A Trojan horse can be attached to a virus, much as a warhead can be attached to a missile.

Trojan horses usually scramble and erase data or the file allocation table. They can begin destruction immediately or wait for weeks or months. They can leave messages and delete themselves from existence once their mission has been accomplished. They are a very potent, destructive weapon.
2600 VAX virus: A virus that replicates itself, sending jobs continuously to the batch queue (where programs line up, waiting to be run). Its only danger is that the queue might overflow.

XmasCard Trojan: First seen on December 9, 1987, a Trojan horse that began in a European academic computer network (Bitnet) and jumped through electronic gateways to five continents and to the internal E-mail system of IBM. In the IBM system, a holiday message appeared in employees' mailboxes, asking them to type "Christmas" on the computer. When they did, it drew a Christmas tree and simultaneously sent copies of itself to all people on that user's personal electronic mail list. See Chapter 3 for further details.

**VIRUS HALL OF FLAME**

Turkey Virus: A virus that causes a turkey to appear on your screen while you are working. As you admire the artwork, it focuses part of the cathode ray beam in your monitor, burning a hole in your screen. Surprise! You are the turkey! Another version of Turkey erases your files while you watch the turkey. First sighted on IBM PC Bulletin Board Systems.

PyroVirus: A virus first sighted in the spring of 1988 in Silicon Valley. It waits until you have left your computer on and unattended for one hour. It then modifies the horizontal scan frequency of your multisync CRT so that it causes the monitor to overheat. This in turn causes the monitor case to melt, creating a real fire danger! So far this virus is only able to carry out its mission with multi-scan terminals.
Computer viruses have been distributed in the following ways:

- **Commercial software**—One of the least likely means of distribution, yet one of the most serious in terms of potentially widespread damage. There has been only one known commercial viral infection so far, and that was the Peace virus on the Macintosh.

- **Bulletin boards**—A very popular means of exchanging programs. Bulletin boards are known to harbor many viruses and are one of the primary means by which viruses are disseminated.

- **Computer stores**—People are always walking in and trying new programs at local computer stores. They are a source of infection, somewhat dependent on the care taken by the management.

- **Friends**—Another very popular means of infection. Friends often infect friends by unknowingly exchanging infected software.
• **Information utilities**—Essentially commercial bulletin boards, they have been known to carry many viruses, including the famous nVIR epidemic (see Chapter 3). Most of the good systems are more vigilant about viruses these days and use vaccines to safeguard their data. Still, be cautious when downloading new software.

• **Public domain software**—Can easily be infected. Make sure you check out programs from this source carefully before you run them!

• **Shareware**—Easily infected. Be careful here as well.

• **Sharing your computer**—Be cautious of new programs being inserted into your system.

• **User groups**—Many user groups are now screening their software before making it available to their members. Before this screening was standard, user groups were a major source of infection.

For the sake of comparison, here are the major methods of biological viral distribution. Do you see the similarities?

<table>
<thead>
<tr>
<th>Methods of Bioviral Distribution</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>The respiratory tract through coughing and sneezing</td>
<td>Measles</td>
</tr>
<tr>
<td>Saliva or nasal fluid contaminating utensils</td>
<td>Common cold</td>
</tr>
<tr>
<td>The fecal-oral route (fecal matter to hands to utensils)</td>
<td>Hepatitis</td>
</tr>
<tr>
<td>Direct skin-to-skin contact</td>
<td>Warts</td>
</tr>
<tr>
<td>Insertion under the skin (as by insects or hypodermic needle)</td>
<td>Yellow fever, hepatitis</td>
</tr>
<tr>
<td>The bite of a rabid animal</td>
<td>Rabies</td>
</tr>
<tr>
<td>Seminal fluid containing blood</td>
<td>AIDS</td>
</tr>
</tbody>
</table>
Primary similarities among computer and biological viruses:

- Needs a host
- Causes general malaise
- Specializes
- Creates both acute and chronic infections

Primary similarities in cures:

- Passive and active barriers strengthening “immune system”
- Isolation

Primary differences:

- Biological viruses have a very complex method of reproduction when compared to computer viruses. The reproductive process for computer viruses is much more orderly and constrained at this point in their evolution.
APPENDIX D
HOW ANTIVIRAL PROGRAMS WORK

TECHNIQUES FOR KEEPING VIRUSES OUT

• Approved program list—Blocks any program not on the list. It does not protect you from approving an infected program, however.

• Approved TSR list—Gives you a warning if any program not on the list attempts to terminate and stay resident.

• Suspicious text search—Displays all text strings in a program. If you see "Ha! Ha! Ha! You’ve been caught!" it’s probably a good idea not to run it.

• Suspicious code search—Looks for suspicious commands and uses of interrupt vectors that do things like low-level reformatting of the hard disk.

• Known virus check—Scans all executable files for known viruses.
TECHNIQUES FOR BLOCKING MALICIOUS TRICKS

- **FAT copy**—Saves a copy of the file allocation table in case a virus manages to damage it in some way. There are several "unformat" programs that already provide this type of protection.

- **CMOS copy**—Saves a copy of CMOS data in case a virus somehow damages it.

- **Hard disk lock**—Temporarily blocks all access to the hard disk while testing suspect software. Most commonly used on IBM AT-type machines.

- **Disk access lockout**—Allows access only through disk operating system file functions. This method prevents reformatting and erasure of the file allocation table.

TECHNIQUES FOR PREVENTING REPLICATION

- **Signature check**—Takes a signature of all approved programs and compares the program with the signature.

- **Run time signature check**—Whenever the DOS loads a program, the program is checked against the signature. If there is not a match, the program doesn’t run.

- **Write protection**—Prevents writing to protected files.

ANTIVIRAL PRODUCTS

New antiviral products are proliferating as viruses become more prevalent among our beloved machines. The following list describes some of the products available to help protect your machine from the viral threat. Some are available commercially, some from user groups and bulletin boards.
- **C-4 Anti-Viral Shield**  
  Developed by John McAfee, Interpath Corporation. Designed to protect against all known viruses attacking the IBM PC environment.

- **Checksum**  
  Usually attached to the end of a program, it is a technique used to see if the size of a program changes.

- **Data Physician** ($199 from Digital Dispatch, 1580 Rice Creek Road, Minneapolis, MN 55432; (612) 571-7400)  
  The oldest known antiviral software, it detects and will sometimes eliminate viruses. Designed for IBM-type and UNIX machines, it makes careful measurement of a computer’s program and data files to detect any alien computer codes. Its components include:
  
  - *Data MD*—This portion of Data Physician creates a list of computer data files to be protected and watches them while the computer is in operation.
  
  - *Antigen*—Attaches itself to an individual computer program and checks it for viruses each time it is used. To remove a virus, *Antigen* erases the bytes of computer data that weren't in the program earlier.
  
  - *Padlock*—Prevents anything from being written on a storage disk unless the computer operator gives permission.

- **Disk Defender** ($199, Elek-tek, 6557 N. Lincoln Ave., Chicago, IL 60645; (800) 621-1269)  
  Developed by Director Technologies, Inc., this product, write-products in hardware all or part of a personal computer hard disk. This protects the operating system and commonly used programs from existing viruses.

- **Disk Watcher** ($79.95, RG Software Systems, Inc., 2300 Computer Ave., Willow Grove, PA 19090; (215) 659-5300)  
  For the IBM PC environment, claims to automatically "pop up" when: a virus is attempting to infect or attack, the disk is full, you ask for a formatting of your hard disk, you try to overwrite a file, you try to print
and the printer isn't ready, or the system time/date is not set.

- **Ferret** (Can be obtained via Macintosh user groups and electronic bulletin boards like CompuServe and MacNET)
  Designed by Larry Nedry and Scott Winders, it notifies an infected user of the date that the Macintosh Scores virus installed itself. It helps determine where and how the virus was picked up.

- ** Forgery Detector**
  An experimental program that analyzes a program's style, in a fashion similar to handwriting analysis. It can detect when foreign code is added to a particular program and can help in determining the author of a virus.

- **Interferon** (Available through Macintosh user groups and electronic bulletin boards)
  Created by Robert Woodhead, a Macintosh shareware program that detects and claims to recognize "signals" that viruses give off when they are present. Interferon was intended to complement the Vaccine program from CE Software of Des Moines, Iowa.

- **KillScores** (Distributed by Howard Upchurch and the Dallas/Ft. Worth Macintosh User's group)
  KillScores unlocks files, disinfects, and leaves files unlocked. It can even repair infected files.

- **Pirate Detector Virus**
  This program keeps track of software duplication. It tells how many copies of a program have been made, and alerts you to illegal or viral program duplication.

- **Protec** ($195 from Sophco, Inc., P.O. Box 7430, Boulder, CO 80306-7430; (800) 922-3001)
  A system of programs for IBM PCs that includes Vaccinate—a virus itself, which infects the host via the Syringe program. It warns the end user (the person using the program as opposed to the one who wrote it)
if a viral infection has occurred. It also includes Canary, a quarantine program. When new files are imported from an unknown source, a user places the Canary program on a diskette with the suspect files. If the Canary dies, a virus program is present.

- **Softlog** (Asky Inc., Milpitas, CA; licensed to corporations in lots of 100 units for $2,400)
  Matches the current size of computer files against their previous size, and thus detects any unauthorized additional material, such as a parasite.

- **Tracer**
  Created by John McAfee of Interpath Corporation. Tracer does low-level access to the interrupt vectors. The virus thinks it is talking to the operating system, but actually it is talking to Tracer.

- **Truss**
  Designed for UNIX systems, it allows the system administrator to examine any process and observe the activities of any user logging in from a remote site. Truss attaches to a login shell (the part of the computer that handles the commands a user needs to log in to the bulletin board). Truss can also freeze a process and allow a debugger more detailed information about the errant process.

- **Vaccine** (Available free on electronic bulletin boards like CompuServe and Genie)
  Created by Don Brown at CE Software, Inc., it enables your Macintosh's operating system to detect alterations to the code of your system files and applications. It requires your permission for any such alterations. If your system is already infected when you install Vaccine, there will be no warning from Vaccine that the virus exists. If Vaccine is installed on a sterile system and the Scores virus is introduced later, Vaccine will only warn of the virus attack; it will not prevent infection.

- **Viralarm System** (Lasertrieve Inc., of Metuchen, NJ; (201) 906-1901)
Using a special program to protect other programs, it creates a software barrier. Works on IBM PCs and most available operating systems.

- **Virex** (HJC Software, P.O. Box 51816, Durham, NC 27717; (919) 490-1277)
  The first commercial antiviral product for the Macintosh, Virex is unique in that it detects the presence of computer viruses and repairs application and system programs damaged by computer viruses by removing the viral code. Combats Scores, nVIR, and the Peace virus. Developed by anti-virus expert Robert Woodhead, the retail price is $99.95.

- **Virus RX**
  Developed by Apple, this is a detection tool to determine whether a system has been infected by the Scores virus, and if so, which applications have been affected. It lists damaged applications, invisible files, altered system files, and altered applications. Virus RX reports different levels of concern from minor warnings to "dangerous" to "fatal" alerts. It first lists damaged applications, those that have not been infected by the virus, but will not work and should probably be removed. Virus RX is available through Apple dealers, AppleLink, and some user-group bulletin boards.

- **Virus Sentry** (a hardware/software solution; $295.00; contact, Wes Thomas (516) 266-1625)
  Software alone is not a viable solution to viruses. A virus can be attached to the antiviral software once it's known how the antiviral software works. Virus Sentry uses software burned into a ROM on a plug-in board for the IBM PC. (If it is not a plug-in board, it won't work right.) It's a cryptoboard that does automatic enciphering of every bit on every disk, automatically, transparently, and invisible to the user. It encodes every bit of data that goes in and out of the computer providing automatic authentication.
APPENDIX E
RECOMMENDED READING FOR VIRUS WATCHERS


GLOSSARY

administrators: Humans who are officially in charge of the financial overhead of large computer systems. The administrators often have no real computer knowledge themselves, preferring to stick with the business end of things. Administrators-vs.-programmers is one of the more popular genre of games played within high-tech domains.

algorithm: A detailed, organized, step-by-step plan for performing an action or solving a problem. When a programmer is preparing to write a program, he or she normally writes algorithms for complex or unfamiliar operations, and these algorithms serve as specifications for the program. The term sometimes also refers to the main part of a complicated program. All computer viruses are programs and hence contain algorithms.

allocation: The assignment of computing resources for some purpose. When a program is started, the operating system allocates an address space to the program. If the program will be writing processed results to a file, the operating system might allocate disk space for this purpose. It could also allocate a disk drive if it requires input from a disk file. Allocations are normally handled by the operating system based upon requests generated by the program. Viruses sometimes erase the allocation table on a disk, equivalent to erasing a book’s table of contents.

anarchist: Someone totally into freedom of everything with no restraints. Anarchists trust the higher will of humans to take care of any necessary organization. A recent invitation to an anarchy picnic read, “Bring your own food, or you will starve. We are not your mother.”

application program: A computer program for a given user that solves a specific problem or performs a specific action, such as producing bills, printing a report, or entering new data into a file. Application programs contrast with system control programs and utility programs, which perform generalized tasks for the benefit of all users of the computer. Application programs are usually the programs that are infected by computer viruses.

assembly language: A programming language in which each action the CPU performs is represented by a symbolic instruction. Assembly language is thus a convenient way of expressing machine language. Writing programs in assembly language is extremely tedious and requires great skill and knowledge of the computer’s inner workings, since every tiny step the machine takes must be spelled out. Consequently, assembly language is not generally considered suitable for application programs. On the other hand, assembly language programs run extremely fast.
occupy relatively little space in memory, and provide absolute control over the computer. For these reasons, assembly language is favored for writing system control programs and other programs where high performance is important. Because assembly language exactly parallels machine language, every CPU type has its own unique assembly language that is seldom compatible with the language of other CPU types.

benign: Opposite of malignant. Refers to a condition of disease, usually cancer, where the tumor is not replicating. A not particularly harmful condition.

bootstrap: 1. The process of starting your computer. In other words, what happens after you hit the "on" switch. It is a term arising from the saying "to pull oneself up by the bootstraps." 2. A program that boots a computer. It is usually in ROM (read-only memory) that gains control as soon as power is applied to the computer. This program loads the operating system from disk into memory, and then turns itself off by passing control to the operating system. The bootstrap program can easily be infected by a virus, commonly called a "boot infector."

bugbuster: A human whose profession is to catch and destroy computer viruses.

central nervous system: The brain and spinal cord, with their nerves and organs. Includes parts of the brain governing consciousness and mental activities; parts of brain, spinal cord, and their sensory and motor nerve fibers controlling skeletal muscles; and organs of the body wall.

chaos: 1. A state of the universe where order and structure has yet to tread. 2. A state experienced by computer and user when a virus strikes.

code: 1. A means of electronically representing letters, digits, and other data symbols with binary numbers, e.g., ASCII. 2. Often used as slang for program.

COM infector: A specific type of virus that infects .COM files.

CompuServe: A computer information and program network available to anyone with a credit card. Has carried virus-infected programs in the past.

computer: A machine that processes information. More specifically, a machine that accepts information, applies procedures defined by a program of instructions furnished in advance, and supplies the results of those procedures.

computer AIDS: A condition where the computer’s defense system is used against itself.

computer bulletin boards: Electronic information systems available to anyone with a personal computer and modem. They are the equivalent of electronic newsletters, existing in practically every small town in America. Each carries its own brand of data, and some of them are known viral carriers.
console: The command position at which an operator controls a computer system. Ordinarily consists of a keyboard, a video display, certain special switches or other controls, and sometimes a printer.

core: Another name for memory, based upon a now obsolete technology that used magnetic rings ("cores") to store bits. The term memory can either refer to internal memory or to an external storage medium; core always means internal memory.

Core War: The game where many humans learned the art and science of digital warfare (see Chapter 2).

cryptography: The science of coding and decoding information. Puzzles, tricks, codes, and clever and Mobius-like thinking are required skills in this ancient human endeavor.

Cyberpast: Stories from computing history. In this book, "Tales From the Cyberpast" brings forth stories of particular relevance to today's outbreak of computer viruses. In a related domain, cybernetics refers to the science of applying principles of human thought and brain processes to computers. Cyborgs are creatures whose bodies and minds are a combination of dynamic carbon and silicon-based integrity patterns. They live in the cyberfuture and like to travel.

CyberPunks: Anarchistic youths whose main allegiance is to the evolution of themselves and their cyberworlds. Living for many hours a day inside their homebrew artificial realities, they don't become enemies unless you attack them first.

Cyberworlds: Computer-generated artificial realities in which the visitor has many interesting and unusual powers. For example, he or she can move around at the speed of light, connecting with other visitors from other times and places. After being in a cyberworld for a while, many visitors are sad to leave, many not wishing to return to a less sublime physical-matter environment. Some believe that the cyberworlds are part of an evolutionary process that is externalizing our nervous systems.

data: Raw information presented to a computer for processing. In general usage, any information under the control of a computer system. This includes programs, operating systems, database files, word-processing files, and so forth. In his most confused moments, St. Silicon, the patron saint of high technology, has been heard muttering, "Data, data everywhere, nor any thought to think!"

data bardo: The place data goes when it is erased or de-rezzed. Didn't you ever wonder what happens to the characters on your screen when you hit the backspace key? After they are de-rezzed, they return to the data bardo to be incarnated as a new alphanumeric character at some future date.

database: A collection of data organized in a file especially for rapid
search and retrieval. In a database system, there are many such files, each one devoted to a particular kind of data element, so that one database holds all the employee names, another all their addresses, another all their dates of birth, and so on. Viruses have been known to erase or modify such files.

data file: See definition for file.

debug: To research and fix bugs (errors) in a program.

debugger: A program that aids in the debugging process. A debugger is interactive and lets a programmer at a video display view his or her program as it executes, stopping it and examining results as they are being produced, tracing events as they occur, and so forth, to enable the programmer to find the errors.

DEC: Digital Equipment Corporation, one of the largest minicomputer companies in the world. Many of today's programmers were weaned on DEC minicomputers like the PDP-8 and PDP-11.

DES: A crypto-code system developed by the National Security Agency for purposes of protecting access to data. A difficult code to crack, it has nevertheless been hacked by computer hackers with some success.

designer virus: Viruses designed for very particular purposes, like infecting programs written by specific individuals. The Scores virus, for example, is designed to infect all programs written by "VULT" or "ERIC."

disk: A high-speed data storage device operating as a peripheral under the control of a computer. A disk is a circular platter that rotates rapidly; in large systems it usually consists of several such platters sharing a common axis (disk pack). Data are recorded as magnetic spots representing bits, written in concentric circles called tracks. A read/write head moves toward or away from the center of the disk, positioning itself over the tracks as directed by the CPU or disk controller. Disk storage is an essential feature of virtually all modern computer systems and the mainstay of data processing. In large computer systems, with dozens to hundreds of disk units, disk storage typically holds billions of characters per unit, with any data being accessible within a few thousandths of a second. Smaller but comparatively powerful disk units are available for microcomputers.

disk crash: What happens when a hard disk fails to function. Millions of data bits can be destroyed forever in a big crash, sending the owner of this data into serious depressive states of awareness. As with the loss of a loved one, the owner first goes through a state of disbelief, followed by rage, then hope, and finally acceptance of the loss. Computer viruses are known for creating these bonus experiences of life.

diskette: A small version of disk storage. A diskette is a single, flexible magnetic disk enclosed within a sealed envelope that inserts
into a diskette drive. It works on the same principle as large disk units, but it holds considerably less data: usually a minuscule million bytes (one megabyte). Diskettes compensate for their lack of capacity by being easily portable and inexpensive. Widely used with mini- and microcomputers, they are also called floppy disks.

disk operating system: A computer operating system that is heavily oriented toward use of disk storage. Commonly called DOS, it can become infected with a virus and spread the infection to programs under its control.

disk swapping: An adult version of baseball card swapping; the exchange of diskettes loaded with data for others. The usual purpose is to increase one's collection of programs and data. This form of data swapping is popular among personal computer users and occurs most frequently at what are known as user group meetings.

distributed databases: The simultaneous use of several geographically separated databases on computers connected by communications facilities to work on a specific problem. Dr. Cohen has hypothesized viruses that can infect such a system.

DNA (Deoxyribonucleic acid): A complex protein of high molecular weight consisting of deoxyribose, phosphoric acid, and four bases. These are arranged as two long chains, which twist around each other to form a double helix joined by bonds between the complementary components. Nucleic acid is present in chromosomes of the nuclei of cells and is the chemical basis of heredity and the carrier of genetic information for most organisms.

dormancy: The period of time when a computer virus is quiet and invisible to the user. The dormancy period is usually controlled by some timing algorithm, and can be likened to the timing mechanism of a time bomb.

electronic mail (E-mail): A software system, usually associated with a computer network, that enables messages to be exchanged among users. An example is interoffice correspondence, in which one user generates a memo and then sends it electronically to one or more users. The recipients are notified by a signal or a message on the screen that they have mail waiting, and they can "pick it up" at their convenience by displaying it, storing it in a disk file, or both. Viruses can infect such a mail system so that every user's personal E-mail correspondence list becomes infected, destroying the whole system within hours of activation. See Chapter 3 for the history of such occurrences to date.

epidemiology: The science concerned with defining and explaining the interrelationships of factors that determine disease frequency and distribution. The scientists of the Center for Disease
Control in Atlanta, Georgia, are the official experts in this field, and researchers at the University of California at Berkeley have an experimental computer disease program.

**evolution**: The force constantly generating the desire to live; a process of orderly and gradual change or development. More generally, any orderly and gradual process of modification whereby a system—whether physical, chemical, social, or intellectual—becomes more highly organized.

**field**: One item of information among several that comprise a record. Data within a computer are organized in this hierarchy: several bits form a character; several characters form a field; several fields form a record; several records form a file; several files form a library.

**file**: A complete, organized collection of information. A file is analogous to a folder in a filing cabinet in that it can be retrieved, opened, processed, closed, moved, and placed back into storage as a unit. The records in a computer file are comparable to documents within the file folder in that a set of records or files are related to each other and are organized in some fashion. Every computer file has a unique name by which it is known and referred to both by humans and by the computer system. A file is a major element within the hierarchy of data organization described under the definition of **field**. A file may contain data, or it may be a computer program. Only the program-type files can be infected by a computer virus at present.

**floppy disk**: A diskette.

**gene**: The basic unit of heredity. Each gene occupies a certain location on a chromosome. Genes are self-producing, ultramicroscopic structures capable under certain circumstances of giving rise to new character. Such a change is called a mutation.

**genetic engineering**: The synthesis, alteration, or repair of genetic material by synthetic means.

**hack**: 1. A quick bit of work that produces what is needed, but not well. 2. An incredibly good, and perhaps very time-consuming, piece of work that produces exactly what is needed. It may also be a brilliant practical joke. The value of the hack varies in proportion to its cleverness, harmlessness, surprise, fame, and appropriate use of technology.

**hacker**: A person who enjoys learning the details of computer systems and how to stretch the systems' capabilities—as opposed to most users of computers, who prefer to learn only the minimum amount necessary.

**hard disk**: A small high-capacity disk storage device for microcomputers. Viruses can cause hard disks to "crash," or lose their data.

**host**: The organism from which a parasite obtains its nourishment.
With computer viruses, the program in which the virus is embedded.

**human mind:** A mysterious process of refining intelligence intimately intertwined and connected with the human body and all the domains it tracks a-, sub-, un-, or preconsciously.

**hypercard:** A nonprogrammer's programming environment allowing the user to create interactive movies using visuals, sound, and text. Works in the Macintosh environment.

**IBM:** International Business Machines, the largest computer company in the world. IBM creates mainframes and microcomputers and is responsible for the IBM PC standard, the most virally infected of all computer types.

**infection:** The state or condition in which the body or part of it is invaded by a pathogenic agent (microorganism or virus), which, under favorable conditions, multiplies and produces effects that are injurious. In high technology, a computer program that is carrying a computer virus.

**interrupt:** A signal sent to the CPU of a computer by a device that requires attention. So called because the CPU must eventually interrupt what it is doing in order to respond. An interrupt is a hardware signal (a pulse carried on a conductor), to which the CPU replies using software called an interrupt handler. It saves the contents of the CPU registers, takes action to satisfy the interrupting device, and then restores the register contents and resumes processing; this is called "servicing an interrupt." Interrupt codes are often used by viruses for doing damage, such as erasing the hard disk.

**laser printer:** Computer output device that uses a laser for printing purposes. Typically, laser printers contain several megabytes of RAM and ROM—memory that can become infected by computer viruses.

**low-level processes:** Processes that allow the user to do things like format the hard disk or read the boot sector, processes that most computer programs never use and couldn't care less about. Such functions are usually reserved for utility programs.

**macrophage:** A wandering, amoebalike cell that has the ability to absorb foreign cells entering the human body. Part of the body's immune system.

**mainframe:** Any large computer system.

**malignant:** Growing worse, resisting treatment, tending or threatening to produce death; said of cancerous growths. A computer virus that is causing damage.

**megabyte:** One million bytes. In computerese, mega- is the square of kilo- (1,024) and therefore actually equals 1,048,576. This is the nearest round number in binary to 1,000,000. The capacity of hard-disk drives and diskettes is usually referred to in terms of
number of megabytes of data the unit is capable of storing. For example, a hard disk may have a 100-megabyte data storage capacity.

memes: Mental genes. Ideas that get passed on from person to person, generation to generation.

memory: The ability and the devices that enable a machine to store information subject to recall. In its broadest sense, memory refers to any hardware capable of serving that end, such as a disk, tape, or semiconductor storage. More specifically, in common usage memory is the high-speed working area associated with the central processor of a computer, where it writes and reads information immediately useful to the tasks in execution. Both the program being run and the data it is currently processing are kept in memory for the duration of the program run. Most computer memories are semiconductor chips called RAM (for random-access memory). Older computers used magnetic rings called core memory; hence the origin of the name of the computer game Core War, a birthplace of computer viruses (see Chapter 2).

mind map: A visual method of tracking information using images or icons representing people, places, or things linked by lines representing relationships between icons. A simple, yet effective means of conveying information, a mind map was successfully used as a media tool in explaining to the jury how a virus was criminally used in the Texas Burleson trial (see Chapter 5).

network: A group of communicating devices, usually terminals, and one or more computers, connected via telephone lines for the purpose of moving information from place to place. The newly emerging network viruses spread the viral infection from one computer or device to the next via the interlinking devices.

NSA: The National Security Agency, a U.S. government organization focused on issues of national security. The agency has developed DES, a crypto-code system for protecting computers, and has an ongoing program studying computer viruses.

nucleus: The vital body in the protoplasm of a cell; the essential agent in growth, metabolism, reproduction, and transmission of characteristics of a cell.

organism: Any living thing, plant or animal. Computer viruses are sometimes referred to as organisms, even though they are simply programs, not living things (so far).

program: A complete set of instructions that tells a computer how to do something. It starts at some point, runs to completion at some other point, and contains all the directions necessary to fulfill its purpose. "To program" means to write computer programs.
programmer: A professional responsible for computer software. The two major occupational groupings are systems programming and applications programming. Systems programmers specialize in the operating system and other system control programs that give overall direction to a data-processing installation. Application programmers design, develop, write, test, and maintain programs that process data.

programming language: A human-oriented language for telling a computer what to do and how to do it. Most programming languages consist of a finite number of English-like verbs (such as PRINT, READ, SAVE, OPEN) and implied verbs (such as +, —, =), which are combined with symbolic data names according to certain rules (the syntax) in order to form statements that make sense to humans. There are many programming languages, including BASIC, Pascal, C, Forth, FORTRAN, COBOL, and ALGOL. Computer viruses are usually programs written in one or more of these languages.

RAM: Random-access memory. A common term for the semiconductor memory of a computer.

reboot: Similar to the "reset," to reboot a computer means to reload the systems software so that the machine makes a fresh start. It is usually done by the computer operator after a problem or glitch has occurred. Information in RAM is usually wiped out during a reboot; most viruses can survive a reboot.

record: One item of information among several that comprise a file. A record is an element within the hierarchy of data organization described under the definition of field.

reference model: A way of thinking; a map you can consult when organizing your thoughts. Most computer security reference models have so far been based on the military model. An example of thinking via the military reference model is the concept of password access. Password access to a computer is based on the analogy of a fence or wall protecting a valued physical asset. When you give the correct password, you are allowed through the electronic logical fence around the computer. Another, more advanced reference model for computers is the human immune system. This model, pioneered by Robert Woodhead, is discussed in Chapter 7.

replicate: To reproduce. For computer viruses, this is a key process. The viruses' survival depends on their ability to replicate successfully. See the Appendixes for details on the replication process.

resources: In the Macintosh environment, applications and programs. Resources can become infected by viruses like Scores; viruses have even been known to create resource files.
ROM: Read-only memory. A semiconductor memory chip whose contents can be read but can be written only by special means.

sector: A subdivision of a track on a disk, constituting a unit of data storage space. On a floppy disk, as an example, the disk might have 77 tracks, with each track subdivided into 26 sectors of 128 bytes of data apiece. An address on a disk consists of the track and sector numbers.

security: A host of activities dedicated to maintaining the well-being of a specified target, be it people or machines. Viruses, be they computer or biological, are perceived by many as threats to our machines and bodies. Therefore, individuals and groups interested in security are studying viruses in great detail at this time. Viruses are a challenge to security experts, because the very nature of the beast suggests that there is no such thing as absolute security.

seek time: The amount of time it takes to position a disk drive's read/write head over a specific track. Seek time varies according to the distance that the head has to move, and is determined by the head stepping rate times the number of tracks that have to be crossed between the point of origin and the destination. By erasing the data on a diskette, computer viruses can put the computer into seek mode for an indefinite period of time. This process has inspired St. Silicon to write the song "On a Clear Disk You Can Seek Forever."

source code: The statements of a computer program written in a programming language.

storage device: A computer peripheral on which information (programs and data) is held in a magnetic form that can be readily read and written by the computer. The main storage devices are disk, drum, tape, and floppy disk drives. Viruses often act by destroying data contained within storage devices.

telemetry probes: Viruses designed to explore computer systems for purposes of collecting data, not unlike the Voyager spacecraft to Jupiter, Saturn, and Uranus.

time bomb: Destructive code attached to a virus that will do damage when specific criteria have been met. These criteria may be anything from a particular date to a specific name that acts as a triggering mechanism.

transmutate: To go beyond normal mutation processes. Sometimes viruses will be changed in ways not predicted by the creator—ways created by situations that may arise in slightly incompatible computer environments.

Trojan horse: Bits of code slipped into an otherwise innocent program. Trojan horses do not replicate as viruses do. Some are written from scratch; some are modified from earlier models.
They can do anything from scrambling data to erasing entire files. The more sophisticated ones delete themselves from existence once they have done their damage.

UNIX: An operating system developed by Bell Laboratories for minicomputers and recently adapted to run on mainframes and microcomputers. UNIX permits several programs to run at the same time and has many programmer development tools. It is one of the most powerful general-purpose systems, and has the advantage of being easily transportable from one system to another. UNIX is widely used on college campuses, and a version of it is being used on the new NeXT machine.

user groups: Regular gatherings of computer users for purposes of exchanging information, programs, and data. A valuable means of keeping up with the latest and greatest in the computer world, the groups are also one of the most likely places to contract a computer virus.

utility program: A general-purpose computer program that performs some activity not specific to any particular application. An example is a reformat program, where the computer's hard disk is erased of data and reordered to receive new data.

vaccine: Medically, a suspension of infectious agents or some part of them, given for the purpose of establishing resistance to an infectious disease. Computer vaccines protect computer systems from getting infected, much like the protection of a male prophylactic.

VAX: A minicomputer designed by Digital Equipment Corporation. One of the first machines to be infected by a virus (see Chapter 1).

virulence: A measure of how fast and well a computer virus can spread from one machine to another. For a comparison of viral infections, see Fred Cohen's data on virulence in Chapter 2.

virus: A software program that can spread from computer to computer and use each infected computer to propagate more copies of itself. Without human intervention, the virus infects the host system, hiding somewhere in the operating system or in an application program. When another computer communicates with the infected host via floppy diskette or telephone lines, the virus slips itself into the other system.

voice mail: An electronic mail system that allows the user to store and forward his or her voice as the means of delivering messages. Such systems have been infected with viruses, causing messages to be lost, scrambled, or both.
Virus! began as a dream. I woke early one morning in a cold sweat and felt compelled to call my dear friend Bonnie Goldstein, an anthropology student at University of California, Santa Cruz. She grasped the concept of the computer virus immediately and inspired me to pursue the creature. After writing down the basic ideas, my next call was to my agent, Bill Gladstone. He wasn't in, so I spoke with Julie Castignola, Bill's fiction editor. She too showed an immediate "hot" response to Virus! and encouraged me to write an outline and sample chapter and get it to her yesterday.

Bernie Shir-Cliff of Contemporary Books, a veteran editor with seasoned insights about the future, picked up the option on Virus! almost immediately from my sample chapter. Bernie has provided me with much feedback over the course of writing Virus!, helping me orient the book to a larger, less technical audience.

On the home front, my deep-felt thanks go to Linda Murmann, who provided a beautiful, loving, and supportive environment for my writing endeavors. Much thanks also to Brett Murmann who further helped me explain the virus from the perspective of baseball in Chapter 9.

And thanks to John Fraiser, a young CyberPunk, who blasted through my door on his Ninja motorcycle one day last summer. John provided me with the first Macintosh on which to write Virus! and was an assistant and friend, lending expert computer guidance as well as viral contacts in the CyberPunk community.
Acknowledgments

I am grateful to the collected members of the Psycho-Physical Space/Time Enlightened Masters Association, whose late-night escapades and insights kept me inspired when the going was tough.

The final stretch was made immeasurably easier by truly dedicated friends such as Jan Lewis, who made her house and computer resources available to me; Craig Larsen, who provided invaluable library research and assisted in the tremendous task of rewriting the second chapter; Duke Houston, who has done everything from providing illustrations to editing to cover art to lending his personal computer; Nancy Collins, who spent five days fine-tuning the manuscript; Tamara Scott, who helped in editing and conducting viral interviews; Jim Rintoule, who provided fax facilities and valuable feedback as a professional book editor; Nancy Schoor, for getting me the hard disk on time; Dan Kottke, for lending me "Cosmic Charlie"; Pat Bailey, for providing political insight into the viral phenomenon; and Taylor Barcroft, who showed up with the right equipment just as things were wrapping up.

To Steve Hammer, where and whenever you are, thanks for bolstering my self-confidence when the going was tough. And to Wes Thomas, thanks for the continuing flow of some of the hottest stuff around, juicy data steeped with the insights that only come from true friendship.

Also a special thanks to communicators Alison Kennedy and the staff of Mondo 2000, Wes Thomas, Amanda Goodenough, Earnie Tello, Brett Leonard, Jemiel Everett, Jane Findley, Jeff Armstrong, Joan McGonagle, Peter Rosencrantz, Michael Moore, John Draper, John Furber, John and Jessica Vaughn, Steveann Auerbach, Jim Karnstedt, and Wayne Lundell for providing me with topical viral information. I am also indebted to my brother, Norman Lundell, who kept me going, and my parents, Bob and Irene Lambert, for their love and viral insights.
Want to know the latest word on viral activity? Curious about the people creating viruses? Interested in meeting the “Sherlock Holmes” of viral investigations?

Then consider ordering:

**VIRUS! The Video**

Our computer animation will show you exactly how viruses work and what can be done about them.

*Order your copy of *VIRUS! The Video* today!*

To order, write to: Allan Lundell
175 Flintlock Ln.
Ben Lomond, CA 95005
"The dangers of viruses are just unbelievable . . . the threat is more serious than most people think."

—Donald Latham, Executive Vice President, Computer Sciences Corporation
Former Assistant Secretary of Defense

Virus! describes a truly frightening scenario for the computer world—the deliberate insertion of false and self-destructive commands into computer programs and core memory.

From the playful invention of Core War and the second-generation games like Bigfoot, Raidar, and Scanner to the exploits of Richard Morris, Jr., whose Internet Worm knocked out six thousand computers nationwide, Virus! is the story of how a handful of hackers and ingenious programmers have opened a Pandora's box we may never be able to close again.

Allan Lundell, former West Coast editor of Byte and Popular Computing, is a journalist specializing in computer science.