

3

Cybersecurity and the Generative Dilemma

In 1988 there were about sixty thousand computers connected to the Internet. Few of them were PCs.¹ Instead, the Net was the province of mainframes, minicomputers, and professional workstations found at government offices, universities, and computer science research centers.² These computers were designed to allow different people to run software on them at the same time from multiple terminals, sharing valuable processor cycles the way adjoining neighbors might share a driveway.³

On the evening of November 2, 1988, many of these computers started acting strangely. Unusual documents appeared in the depths of their file systems, and their system logs recorded activities unrelated to anything the computers' regular users were doing. The computers also started to slow down. An inventory of the running code on the machines showed a number of rogue programs demanding processor time. Concerned administrators terminated these foreign programs, but they reappeared and then multiplied. Within minutes, some computers started running so slowly that their keepers were unable to investigate further. The machines were too busy attending to the wishes of the mysterious software.

System administrators discovered that renegade code was spreading through the Internet from one machine to another. In response, some unplugged their computers from the rest of the world, inoculating them from further attacks but sacrificing all communication. Others kept their machines plugged in and, working in groups, figured out how to kill the invading software and protect their machines against re-infection.

The software—now commonly thought of as the first Internet worm—was traced to a twenty-three-year-old Cornell University graduate student named Robert Tappan Morris, Jr. He had launched it by infecting a machine at MIT from his terminal in Ithaca, New York.⁴ The worm identified other nearby computers on the Internet by rifling through various electronic address books found on the MIT machine.⁵ Its purpose was simple: to transmit a copy of itself to the machines, where it would there run alongside existing software—and repeat the cycle.⁶

An estimated five to ten percent of all Internet-connected machines had been compromised by the worm in the span of a day. Gene Spafford of Purdue University called it an “attack from within.”⁷ The program had accessed the machines by using a handful of digital parlor tricks—tricks that allowed it to run without having an account on the machine. Sometimes it exploited a flaw in a commonly used e-mail transmission program running on the victimized computers, rewriting the program to allow itself in. Other times it simply guessed users’ passwords.⁸ For example, a user named jsmith often chose a password of . . . jsmith. And if not, the password was often obvious enough to be found on a list of 432 common passwords that the software tested at each computer.⁹

When asked why he unleashed the worm, Morris said he wanted to count how many machines were connected to the Internet. (Proprietary networks were designed to keep track of exactly how many subscribers they had; the simple Internet has no such mechanism.) Morris’s program, once analyzed, accorded with this explanation, but his code turned out to be buggy. If Morris had done it right, his program would not have slowed down its infected hosts and thereby not drawn attention to itself. It could have remained installed for days or months, and it could have quietly performed a wide array of activities other than simply relaying a “present and accounted for” message to Morris’s designated home base to assist in his digital nose count.

The university workstations of 1988 were generative: their users could write new code for them or install code written by others. The Morris worm was the first large-scale demonstration of a vulnerability of generativity: even in the

custody of trained administrators, such machines could be commandeered and reprogrammed, and, if done skillfully, their users would probably not even notice. The opportunity for such quick reprogramming vastly expanded as these workstations were connected to the Internet and acquired the capacity to receive code from afar.

Networked computers able to retrieve and install code from anyone else on the network are much more flexible and powerful than their applianceized counterparts would be. But this flexibility and power are not without risks. Whether through a sneaky vector like the one Morris used, or through the front door, when a trusting user elects to install something that looks interesting but without fully inspecting it and understanding what it does, opportunities for accidents and mischief abound. Today's generative PCs are in a similar but more pronounced bind, one characterized by faster networks, more powerful processors, and less-skilled users.

A MILD AUTOIMMUNE REACTION

The no-longer-theoretical prospect that a large swath of Internet-connected computers could be compromised, and then contribute to the attack of others, created a stir. But to most, the Morris attack remained more a curiosity than a call to arms. Keith Bostic of the University of California–Berkeley computer science department described in a retrospective news account the fun of trying to puzzle out the problem and defeat the worm. “For us it was a challenge. . . . It wasn't a big deal.”¹⁰

Others perceived the worm as a big deal but did little to fix the problem. The mainstream media had an intense but brief fascination with the incident.¹¹ A professional organization for computer scientists, the Association for Computing Machinery, devoted an issue of its distinguished monthly journal to the worm,¹² and members of Congress requested a report from its research arm, the U.S. General Accounting Office (GAO).¹³

The GAO report noted some ambiguities and difficulties in U.S. law that might make prosecution of worm- and virus-makers burdensome,¹⁴ and called for the creation of a government committee to further consider Internet security, staffed by representatives of the National Science Foundation, the Department of Defense, and other agencies that had helped fund the Internet's development and operation.¹⁵ At the time it was thought that the Internet would evolve into a “National Research Network,” much larger and faster, but still used primarily by educational and other noncommercial entities in loose coor-

dination with their U.S. government sponsors.¹⁶ The most tangible result from the government inquiry was a Defense Department–funded program at Carnegie Mellon University called CERT/CC, the “Computer Emergency Response Team Coordination Center.” It still exists today as a clearinghouse for information about viruses and other network threats.¹⁷

Cornell impaneled a commission to analyze what had gone wrong. Its report exonerated the university from institutional responsibility for the worm and laid the blame solely on Morris, who had, without assistance or others’ knowledge, engaged in a “juvenile act” that was “selfish and inconsiderate.”¹⁸ It rebuked elements of the media that had branded Morris a hero for exposing security flaws in dramatic fashion, noting that it was well known that the computers’ Unix operating systems had many security flaws, and that it was no act of “genius” to exploit such weaknesses.¹⁹ The report called for a university-wide committee to advise the university on technical security standards and another to write a campus-wide acceptable use policy.²⁰ It described consensus among computer scientists that Morris’s acts warranted some form of punishment, but not “so stern as to damage permanently the perpetrator’s career.”²¹ That is just how Morris was punished. He apologized, and criminal prosecution for the act earned him three years of probation, four hundred hours of community service, and a \$10,050 fine.²² His career was not ruined. Morris transferred from Cornell to Harvard, founded a dot-com startup with some friends in 1995, and sold it to Yahoo! in 1998 for \$49 million.²³ He finished his degree and is now a tenured professor at MIT.²⁴

As a postmortem to the Morris worm incident, the Internet Engineering Task Force, the far-flung, unincorporated group of engineers who work on Internet standards and who have defined its protocols through a series of formal “request for comments” documents, or RFCs, published informational RFC 1135, titled “The Helminthiasis of the Internet.”²⁵ RFC 1135 was titled and written with whimsy, echoing reminiscences of the worm as a fun challenge. The RFC celebrated that the original “old boy” network of “UNIX system wizards” was still alive and well despite the growth of the Internet: teams at university research centers put their heads together—on conference calls as well as over the Internet—to solve the problem.²⁶ After describing the technical details of the worm, the document articulated the need to instill and enforce ethical standards as new people (mostly young computer scientists like Morris) signed on to the Internet.²⁷

These reactions to the Morris worm may appear laughably inadequate, an unwarranted triumph of the principles of procrastination and trust described

earlier in this book. Urging users to patch their systems and asking hackers to behave more maturely might, in retrospect, seem naïve. To understand why these were the only concrete steps taken to prevent another worm incident—even a catastrophically destructive one—one must understand just how deeply computing architectures, both then and now, are geared toward flexibility rather than security, and how truly costly it would be to retool them.

THE GENERATIVE TRADE-OFF

To understand why the Internet-connected machines infected by the Morris worm were so vulnerable, consider the ways in which proprietary networks were more easily secured.

The U.S. long distance telephone network of the 1970s was intended to convey data between consumers in the form of telephone conversations. A group of hackers discovered that a tone at a frequency of 2,600 hertz sent over a telephone line did not reach the other side, but instead was used by the phone company to indicate to itself that the line was idle.²⁸ For example, the tone could be used by a pay phone to tell network owner AT&T that it was ready for the next call. It was not intended for customers to discover, much less use. As fortune would have it, a children's toy whistle packaged as a prize in boxes of Cap'n Crunch cereal could, when one hole was covered, generate a shrill tone at exactly that frequency.²⁹

People in the know could then dial toll-free numbers from their home phones, blow the whistle to clear but not disconnect the line, and then dial a new, non-toll-free number, which would be connected without charge.³⁰ When this vulnerability came to light, AT&T was mortified, but it was also able to reconfigure the network so that the 2,600 hertz tone no longer controlled it.³¹ Indeed, the entire protocol of *in-band* signaling could be and was eliminated. Controlling the network now required more than just a sound generated at a telephone mouthpiece on one end or the other. Data to be sent between customers and instructions intended to affect the network could be separated from one another, because AT&T's centralized control structure made it possible to separate the transfer of data (that is, conversations) between customers from instructions that affected network operations.³²

The proprietary consumer networks of the 1980s used similar approaches to prevent network problems. No worm could spread on CompuServe in the same manner as Morris's, because CompuServe already followed the post-Cap'n Crunch rule: do not let the paths that carry data also carry code. The

consumer computers attached to the CompuServe network were configured as mere “dumb terminals.” They exchanged data, not programs, with CompuServe. Subscribers browsed weather, read the news, and posted messages to each other. Subscribers were not positioned easily to run software encountered through the CompuServe network, although on occasion and in very carefully labeled circumstances they could download new code to run on their generative PCs separately from their dumb terminal software.³³ The mainframe computers at CompuServe with which those dumb terminals communicated existed out of view, ensuring that the separation between users and programmers was strictly enforced.³⁴

These proprietary networks were not user-programmable but instead relied on centralized feature rollouts performed exclusively by their administrators. The networks had only the features their owners believed would be economically viable. Thus, the networks evolved slowly and with few surprises either good or bad. This made them both secure and sterile in comparison to generative machines hooked up to a generative network like the Internet.

Contrary to CompuServe’s proprietary system, the Internet of 1988 had no control points where one could scan network traffic for telltale wormlike behaviors and then stop such traffic. Further, the Morris worm really was not perceived as a *network* problem, thanks to the intentional conceptual separation of network and endpoint. The Morris worm used the network to spread but did not attack it beyond slowing it down as the worm multiplied and continued to transmit itself. The worm’s targets were the network’s endpoints: the computers attached to it. The modularity that inspired the Internet’s design meant that computer programming enthusiasts could write software for computers without having to know anything about the network that would carry the resulting data, while network geeks could devise new protocols with a willful ignorance of what programs would run on the devices hooked up to it, and what data would result from them. Such ignorance may have led those overseeing network protocols and operation unduly to believe that the worm was not something they could have prevented, since it was not thought to be within their design responsibility.

In the meantime, the endpoint computers could be compromised because they were general-purpose machines, running operating systems for which outsiders could write executable code.³⁵ Further, the operating systems and applications running on the machines were not perfect; they contained flaws that rendered them more accessible to uninvited code than their designers intended.³⁶ Even without such flaws, the machines were intentionally designed

to be operated at a distance, and to receive and run software sent from a distance. They were powered on and attached to the network continuously, even when not in active use by their owners. Moreover, many administrators of these machines were lazy about installing available fixes to known software vulnerabilities, and often utterly predictable in choosing passwords to protect entry to their computer accounts.³⁷ Since the endpoint computers infected by the worm were run and managed by disparate groups who answered to no single authority for their use, there was no way to secure them all against attack.³⁸

A comparison with its proprietary network and information appliance counterparts, then, reveals the central security dilemma of yesterday's Internet that remains with us today: the proprietary networks did not have the Cap'n Crunch problem, and the Internet and its connected machines do. On the Internet, the channels of communication are also channels of control.³⁹ There is no appealing fix of the sort AT&T undertook for its phone network. If one applies the post-Cap'n Crunch rule and eliminates the ability to control PCs via the Internet—or the ability of the attached computers to initiate or accept such control—one has eliminated the network's generative quality. Such an action would not merely be inconvenient, it would be incapacitating. Today we need merely to click to install new code from afar, whether to watch a video newscast embedded within a Web page or to install whole new applications like word processors or satellite image browsers. That quality is essential to the way in which we use the Internet.

It is thus not surprising that there was little impetus to institute changes in the network in response to the Morris worm scare, even though Internet-connected computers suffered from a fundamental security vulnerability. The decentralized, nonproprietary ownership of the Internet and the computers it linked made it difficult to implement any structural revisions to the way it functioned, and, more important, it was simply not clear what curative changes could be made that did not entail drastic, wholesale, purpose-altering changes to the very fabric of the Internet. Such changes would be so wildly out of proportion with the perceived level of threat that the records of postworm discussion lack any indication that they were even considered.

As the next chapter will explore, generative systems are powerful and valuable, not only because they foster the production of useful things like Web browsers, auction sites, and free encyclopedias, but also because they can allow an extraordinary number of people to express themselves in speech, art, or code and to work with other people in ways previously not possible. These characteristics can make generative systems very successful even though they lack cen-

tral coordination and control. That success draws more participants to the generative system. Then it stalls.

Generative systems are built on the notion that they are never fully complete, that they have many uses yet to be conceived of, and that the public can be trusted to invent and share good uses. Multiplying breaches of that trust can threaten the very foundations of the generative system. A hobbyist computer that crashes might be a curiosity, but when a home or office PC with years' worth of vital correspondence and papers is compromised it can be a crisis. As such events become commonplace throughout the network, people will come to prefer security to generativity. If we can understand how the generative Internet and PC have made it as far as they have without true crisis, we can predict whether they can continue, and what would transpire following a breaking point. There is strong evidence that the current state of affairs is not sustainable, and what comes next may exact a steep price in generativity.

AN UNTENABLE STATUS QUO

The Internet and its generative machines have muddled along pretty well since 1988, despite the fact that today's PCs are direct descendants of that era's unsecured workstations. In fact, it is striking how *few* truly disruptive security incidents have happened since 1988. Rather, a network designed for communication among academic and government researchers appeared to scale beautifully as hundreds of millions of new users signed on during the 1990s, a feat all the more impressive when one considers how demographically different the new users were from the 1988 crowd. However heedless the network administrators of the late '80s were to good security practice, the mainstream consumers of the '90s were categorically worse. Few knew how to manage or code their generative PCs, much less how to rigorously apply patches or observe good password security.

The threat presented by bad code has slowly but steadily increased since 1988. The slow pace, which has let it remain a back-burner issue, is the result of several factors which are now rapidly attenuating. First, the computer scientists of 1988 were right that the hacker ethos frowns upon destructive hacking.⁴⁰ Morris's worm did more damage than he intended, and for all the damage it did do, the worm *had no payload other than itself*. Once a system was compromised by the worm it would have been trivial for Morris to have directed the worm to, for instance, delete as many files as possible.⁴¹ Morris did not do this, and the overwhelming majority of viruses that followed in the 1990s reflected similar

authorial forbearance. In fact, the most well-known viruses of the '90s had completely innocuous payloads. For example, 2004's Mydoom spread like wildfire and affected connectivity for millions of computers around the world. Though it reputedly cost billions of dollars in lost productivity, the worm did not tamper with data, and it was programmed to stop spreading at a set time.⁴²

The bad code of the '90s merely performed attacks for the circular purpose of spreading further, and its damage was measured by the effort required to eliminate it at each site of infection and by the burden placed upon network traffic as it spread, rather than by the number of files it destroyed or by the amount of sensitive information it compromised. There are only a few exceptions. The infamous Lovebug worm, released in May 2000, caused the largest outages and damage to Internet-connected PCs to date.⁴³ It affected more than just connectivity: it overwrote documents, music, and multimedia files with copies of itself on users' hard drives. In the panic that followed, software engineers and antivirus vendors mobilized to defeat the worm, and it was ultimately eradicated.⁴⁴ Lovebug was an anomaly. The few highly malicious viruses of the time were otherwise so poorly coded that they failed to spread very far. The Michelangelo virus created sharp anxiety in 1992, when antivirus companies warned that millions of hard drives could be erased by the virus's dangerous payload. It was designed to trigger itself on March 6, the artist's birthday. The number of computers actually affected was only in the tens of thousands—it spread only through the pre-Internet exchange of infected floppy diskettes—and it was soon forgotten.⁴⁵ Had Michelangelo's birthday been a little later in the year—giving the virus more time to spread before springing—it could have had a much greater impact. More generally, malicious viruses can be coded to avoid the problems of real-world viruses whose virulence helps stop their spread. Some biological viruses that incapacitate people too quickly can burn themselves out, destroying their hosts before their hosts can help them spread further.⁴⁶ Human-devised viruses can be intelligently designed—fine-tuned to spread before biting, or to destroy data within their hosts while still using the host to continue spreading.

Another reason for the delay of truly destructive malware is that network operations centers at universities and other institutions became more professionalized between the time of the Morris worm and the advent of the mainstream consumer Internet. For a while, most of the Internet's computers were staffed by professional administrators who generally heeded admonitions to patch regularly and scout for security breaches. They carried beepers and were prepared to intervene quickly in the case of an intrusion. Less adept mainstream con-

sumers began connecting unsecured PCs to the Internet in earnest only in the mid-1990s. At first their machines were hooked up only through transient dial-up connections. This greatly limited both the amount of time per day during which they were exposed to security threats, and the amount of time that, if compromised and hijacked, they would themselves contribute to the problem.⁴⁷

Finally, there was no business model backing bad code. Programs to trick users into installing them, or to bypass users entirely and just sneak onto the machine, were written only for fun or curiosity, just like the Morris worm. There was no reason for substantial financial resources to be invested in their creation, or in their virulence once created. Bad code was more like graffiti than illegal drugs. Graffiti is comparatively easier to combat because there are no economic incentives for its creation.⁴⁸ The demand for illegal drugs creates markets that attract sophisticated criminal syndicates.

Today each of these factors has substantially diminished. The idea of a Net-wide set of ethics has evaporated as the network has become so ubiquitous. Anyone is allowed online if he or she can find a way to a computer and a connection, and mainstream users are transitioning to always-on broadband. In July 2004 there were more U.S. consumers on broadband than on dial-up,⁴⁹ and two years later, nearly twice as many U.S. adults had broadband connections in their homes than had dial-up.⁵⁰ PC user awareness of security issues, however, has not kept pace with broadband growth. A December 2005 online safety study found 81 percent of home computers to be lacking first-order protection measures such as current antivirus software, spyware protection, and effective firewalls.⁵¹ The Internet's users are no longer skilled computer scientists, yet the PCs they own are more powerful than the fastest machines of the 1980s. Because modern computers are so much more powerful, they can spread malware with greater efficiency than ever.

Perhaps most significantly, there is now a business model for bad code—one that gives many viruses and worms payloads for purposes other than simple reproduction.⁵² What seemed truly remarkable when it was first discovered is now commonplace: viruses that compromise PCs to create large “botnets” open to later instructions. Such instructions have included directing the PC to become its own e-mail server, sending spam by the thousands or millions to e-mail addresses harvested from the hard disk of the machine itself or gleaned from Internet searches, with the entire process typically unnoticeable to the PC's owner. At one point, a single botnet occupied 15 percent of Yahoo's entire search capacity, running random searches on Yahoo to find text that could be

inserted into spam e-mails to throw off spam filters.⁵³ One estimate pegs the number of PCs involved in such botnets at 100 to 150 million, or a quarter of all the computers on the Internet as of early 2007,⁵⁴ and the field is expanding: a study monitoring botnet activity in 2006 detected, on average, the emergence of 1 million new bots per month.⁵⁵ But as one account pulling together various guesses explains, the science is inexact:

MessageLabs, a company that counts spam, recently stopped counting bot-infected computers because it literally could not keep up. It says it quit when the figure passed about 10 million a year ago. Symantec Corp. recently said it counted 6.7 million active bots during an Internet scan. Since all bots are not active at any given time, the number of infected computers is likely much higher. And Dave Dagon, who recently left Georgia Tech University to start a bot-fighting company named Damballa, pegs the number at closer to 30 million. The firm uses a “capture, mark, and release,” strategy borrowed from environmental science to study the movement of bot armies and estimate their size.

“It’s like asking how many people are on the planet, you are wrong the second you give the answer. . . . But the number is in the tens of millions,” Dagon said. “Had you told me five years ago that organized crime would control 1 out of every 10 home machines on the Internet, I would have not have believed that. And yet we are in an era where this is something that is happening.”⁵⁶

In one notable experiment conducted in the fall of 2003, a researcher connected a PC to the Internet that simulated running an “open proxy”—a condition in which many PC users unintentionally find themselves.⁵⁷ Within nine hours, spammers’ worms located the computer and began attempting to commandeer it. Sixty-six hours later the researcher had recorded attempts to send 229,468 distinct messages to 3,360,181 would-be recipients.⁵⁸ (The researcher’s computer pretended to deliver on the spam, but in fact threw it away.) Such zombie computers were responsible for more than 80 percent of the world’s spam in June 2006, and spam in turn accounted for an estimated 80 percent of the world’s total e-mail.⁵⁹ North American PCs led the world in December 2006, producing approximately 46 percent of the world’s spam.⁶⁰ That spam produces profit, as a large enough number of people actually buy the items advertised or invest in the stocks touted.⁶¹

Botnets can also be used to launch coordinated attacks on a particular Internet endpoint. For example, a criminal can attack an Internet gambling Web site and then extort payment to make the attacks stop. The going rate for a botnet to launch such an attack is reputed to be about \$50,000 per day.⁶² Virus mak-

ers compete against each other to compromise PCs exclusively, some even using their access to install hacked versions of antivirus software on victim computers so that they cannot be poached away by other viruses.⁶³ The growth of virtual worlds and massively multiplayer online games provides another economic incentive for virus creators. As more and more users log in, create value, and buy and sell virtual goods, some are figuring out ways to turn such virtual goods into real-world dollars. Viruses and phishing e-mails target the acquisition of gaming passwords, leading to virtual theft measured in real money.⁶⁴

The economics is implacable: viruses are now valuable properties, and that makes for a burgeoning industry in virus making where volume matters. Well-crafted worms and viruses routinely infect vast swaths of Internet-connected personal computers. In 2004, for example, the Sasser worm infected more than half a million computers in three days. The Sapphire/Slammer worm in January 2003 went after a particular kind of Microsoft server and infected 90 percent of those servers, about 120,000 of them, within *ten minutes*. Its hijacked machines together were performing fifty-five million searches per second for new targets just three minutes after the first computer fell victim to it. The sobig.f virus was released in August 2003 and within two days accounted for approximately 70 percent of all e-mail in the world, causing 23.2 million virus-laden e-mails to arrive on AOL's doorstep alone. Sobig was designed by its author to expire a few weeks later.⁶⁵ In May 2006 a virus exploiting a vulnerability in Microsoft Word propagated through the computers of the U.S. Department of State in eastern Asia, forcing the machines to be taken offline during critical weeks prior to North Korea's missile tests.⁶⁶

Antivirus companies receive about two reports a minute of possible new viruses in the wild, and have abandoned individual review by staff in favor of automated sorting of viruses to investigate only the most pressing threats.⁶⁷ Antivirus vendor Eugene Kaspersky of Kaspersky Labs told an industry conference that antivirus vendors "may not be able to withstand the onslaught."⁶⁸ Another vendor executive said more directly: "I think we've failed."⁶⁹

CERT/CC's malware growth statistics confirm the anecdotes. The organization began documenting the number of attacks—called "incidents"—against Internet-connected systems from its founding in 1988, as reproduced in Figure 3.1.

The increase in incidents since 1997 has been roughly geometric, doubling each year through 2003. In 2004, CERT/CC announced that it would no longer keep track of the figure, since attacks had become so commonplace and widespread as to be indistinguishable from one another.⁷⁰ IBM's Internet Se-

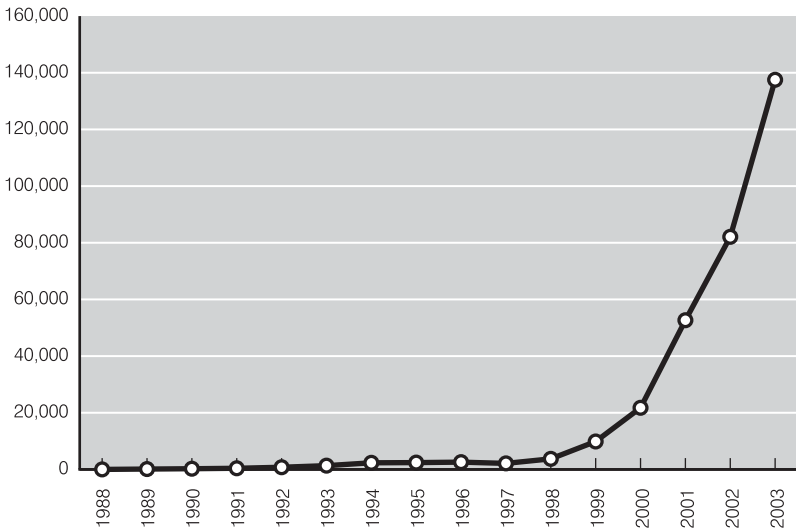


Figure 3.1 Number of security incidents reported to CERT/CC, 1988–2003. *Source:* CERT Coordination Center, CERT/CC Statistics 1988–2005, <http://www.cert.org/stats#incidents>.

security Systems reported a 40 percent increase in Internet vulnerabilities—situations in which a machine was compromised, allowing access or control by attackers—between 2005 and 2006.⁷¹ Nearly all of those vulnerabilities could be exploited remotely, and over half allowed attackers to gain full access to the machine and its contents.⁷² Recall that at the time of the Morris worm there were estimated to be 60,000 distinct computers on the Internet. In July 2006 the same metrics placed the count at over 439 million.⁷³ Worldwide there were approximately 1.1 billion e-mail users in 2006.⁷⁴ By one credible estimate, there will be over 290 million PCs in use in the United States by 2010 and 2 billion PCs in use worldwide by 2011.⁷⁵ In part because the U.S. accounts for 18 percent of the world’s computer users, it leads the world in almost every type of commonly measured security incident (Table 3.1, Figure 3.2).⁷⁶

These numbers show that viruses are not simply the province of computing backwaters, away from the major networks where there has been time to develop effective countermeasures and best practices. Rather, the war is being lost across the board. Operating system developers struggle to keep up with providing patches for newly discovered computer vulnerabilities. Patch development time increased throughout 2006 for all of the top operating system providers (Figure 3.3).⁷⁷

Table 3.1. Rankings of malicious activity by country

Country	Malicious Code	Spam Hosts	Command and Control Services	Phishing Hosts	Bots	Attacks
United States	1	1	1	1	2	1
China	3	2	4	8	1	2
Germany	7	3	3	2	4	3
France	9	4	14	4	3	4
United Kingdom	4	13	9	3	6	6
South Korea	12	9	2	9	11	9
Canada	5	23	5	7	10	5
Spain	13	5	15	16	5	7
Taiwan	8	11	6	6	7	11
Italy	2	8	10	14	12	10

Source: SYMANTEC CORP., SYMANTEC INTERNET SECURITY THREAT REPORT: TRENDS FOR JULY–DECEMBER 06, at 9 (2007) [hereinafter SYMANTEC INTERNET SECURITY THREAT REPORT], http://eval.symantec.com/mktginfo/enterprise/white_papers/ent-whitepaper_internet_security_threat_report_xi_03_2007.en-us.pdf.

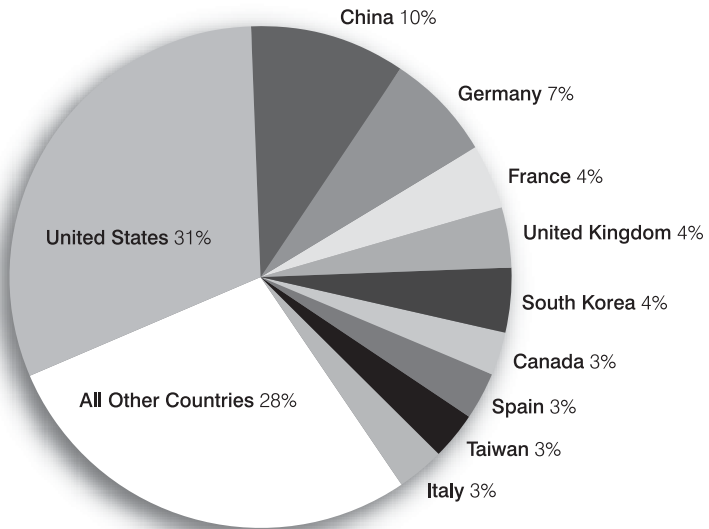


Figure 3.2 Countries as a percentage of all detected malicious activity. Source: SYMANTEC INTERNET SECURITY THREAT REPORT at 26.

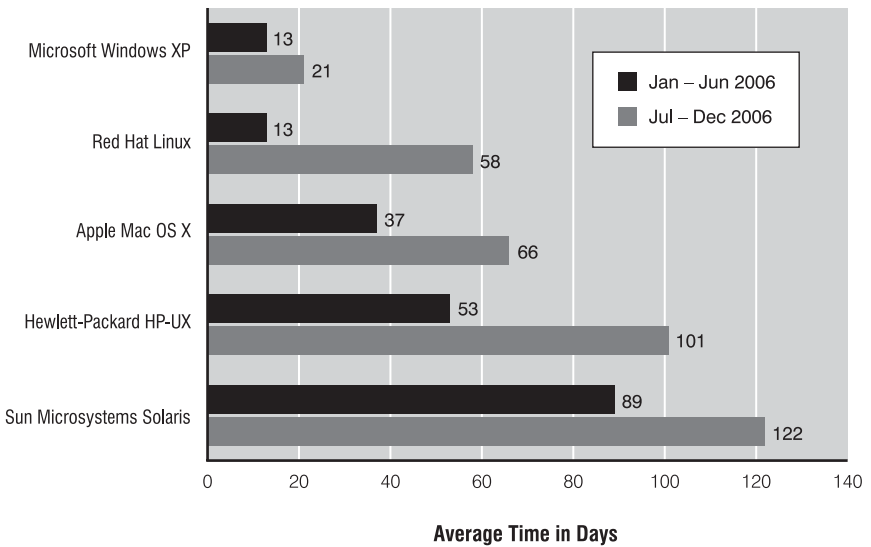


Figure 3.3 Patch development time by operating system. *Source:* SYMANTEC INTERNET SECURITY THREAT REPORT at 39–40.

Antivirus researchers and firms require extensive coordination efforts just to agree on a naming scheme for viruses as they emerge—much less a strategy for battling them.⁷⁸ Today, the idea of casually cleaning a virus off of a PC once it has been infected has been abandoned. When computers are compromised, users are now typically advised to completely reinstall everything on them—either losing all their data or laboriously figuring out what to save and what to excise. For example, in 2007, some PCs at the U.S. National Defense University fell victim to a virus. The institution shut down its network servers for two weeks and distributed new laptops to instructors, because “the only way to ensure the security of the systems was to replace them.”⁷⁹

One Microsoft program manager colorfully described the situation: “When you are dealing with rootkits and some advanced spyware programs, the only solution is to rebuild from scratch. In some cases, there really is no way to recover without nuking the systems from orbit.”⁸⁰

In the absence of such drastic measures, a truly “mal” piece of malware could be programmed to, say, erase hard drives, transpose numbers inside spreadsheets randomly, or intersperse nonsense text at random intervals in Word documents found on infected computers—and nothing would stand in the way.

A massive number of always-on powerful PCs with high-bandwidth con-

nections to the Internet and run by unskilled users is a phenomenon new to the twenty-first century.⁸¹ This unprecedented set of circumstances leaves the PC and the Internet vulnerable to across-the-board compromise. If one carries forward the metaphor of “virus” from its original public health context,⁸² today’s viruses are highly and near-instantly communicable, capable of causing worldwide epidemics in a matter of hours.⁸³ The symptoms may reveal themselves to users upon infection or they may lie in remission, at the whim of the virus author, while the virus continues to spread. Even fastidiously protected systems can suffer from a widespread infection, since the spread of a virus can disrupt network connectivity. And, as mentioned earlier, sometimes viruses are programmed to attack a particular network host by sending it a barrage of requests. Summed across all infected machines, such a distributed denial of service attack can ruin even the most well-connected and well-defended server, even if the server itself is not infected.

The compounded threat to the system of generative PCs on a generative network that arises from the system’s misuse hinges on both the ability of a few malicious experts to bring down the system and this presence of a large field of always-connected, easily exploited computers. Scholars like Paul Ohm caution that the fear inspired by anecdotes of a small number of dangerous hackers should not provide cause for overbroad policy, noting that security breaches come from many sources, including laptop theft and poor business practices.⁸⁴ Ohm’s concern about regulatory overreaction is not misplaced. Nonetheless, what empirical data we have substantiate the gravity of the problem, and the variety of ways in which modern mainstream information technology can be subverted does not lessen the concern about any given vector of compromise. Both the problem and the likely solutions are cause for concern.

Recognition of the basic security problem has been slowly growing in Internet research communities. Nearly two-thirds of academics, social analysts, and industry leaders surveyed by the Pew Internet & American Life Project in 2004 predicted serious attacks on network infrastructure or the power grid in the coming decade.⁸⁵ Though few appear to employ former U.S. cybersecurity czar Richard Clarke’s evocative language of a “digital Pearl Harbor,”⁸⁶ experts are increasingly aware of the vulnerability of Internet infrastructure to attack.⁸⁷

When will we know that something truly has to give? There are at least two possible models for a fundamental shift in our tolerance of the status quo: a collective watershed security moment, or a more glacial death of a thousand cuts. Both are equally threatening to the generativity of the Internet.

A WATERSHED SCENARIO

Suppose that a worm is released that exploits security flaws both in a commonly used Web server and in a Web browser found on both Mac and Windows platforms. The worm quickly spreads through two mechanisms. First, it randomly knocks on the doors of Internet-connected machines, immediately infecting vulnerable Web servers that answer the knock. Unwitting consumers, using vulnerable Internet browsers, visit the infected servers, which infect users' computers. Compromised machines become zombies, awaiting direction from the worm's author. The worm asks its zombies to look for other nearby machines to infect for a day or two and then tells the machines to erase their own hard drives at the stroke of midnight, adjusting for time zones to make sure the collective crash takes place at the same time around the globe.

This is not science fiction. It is merely another form of the Morris episode, a template that has been replicated countless times since, so often that those who run Web servers are often unconcerned about exploits that might have crept into their sites. Google and StopBadware.org, which collaborate on tracking and eliminating Web server exploits, report hundredfold increases in exploits between August 2006 and March 2007. In February 2007, Google found 11,125 infected servers on a web crawl.⁸⁸ A study conducted in March 2006 by Google researchers found that out of 4.5 million URLs analyzed as potentially hosting malicious code, 1.15 million URLs were indeed distributing malware.⁸⁹ Combine one well-written worm of the sort that can penetrate firewalls and evade antivirus software with one truly malicious worm-writer, and we have the prospect of a panic-generating event that could spill over to the real world: no check-in at some airline counters using Internet-connected PCs; no overnight deliveries or other forms of package and letter distribution; no payroll software producing paychecks for millions of workers; the elimination, release, or nefarious alteration of vital personal records hosted at medical offices, schools, town halls, and other data repositories that cannot afford a full-time IT staff to perform backups and ward off technological demons. Writing and distributing such a worm could be a tempting act of information warfare by any of the many enemies of modernity—*asymmetric warfare* at that, since the very beliefs that place some enemies at odds with the developed world may lead them to rely less heavily on modern IT themselves.

A GLACIAL SHIFT

The watershed scenario is plausible, but a major malware catastrophe depends on just the right combination of incentives, timing, and luck. Truly malicious foes like terrorists may see Internet-distributed viruses as damaging but refrain from pursuing them because they are not terror-inducing: such events simply do not create fear the way that lurid physical attacks do. Hackers who hack for fun still abide by the ethic of doing no or little harm by their exploits. And those who hack for profit gain little if their exploits are noticed and disabled, much less if they should recklessly destroy the hosts they infect.

Hacking a machine to steal and exploit any personal data within is currently labor-intensive; credit card numbers can be found more easily through passive network monitoring or through the distribution of phishing e-mails designed to lure people voluntarily to share sensitive information.⁹⁰ (To be sure, as banks and other sensitive destinations increase security on their Web sites through such tools as two-factor authentication, hackers may be more attracted to PC vulnerabilities as a means of compromise.⁹¹ A few notable instances of bad code directed to this purpose could make storing data on one's PC seem tantamount to posting it on a public Web site.)

Finally, even without major security innovations, there are incremental improvements made to the growing arsenals of antivirus software, updated more quickly thanks to always-on broadband and boasting ever more comprehensive databases of viruses. Antivirus software is increasingly being bundled with new PCs or built into their operating systems.

These factors defending us against a watershed event are less effective against the death of a thousand cuts. The watershed scenario, indeed any threat following the Morris worm model, is only the most dramatic rather than most likely manifestation of the problem. Good antivirus software can still stop obvious security threats, but much malware is no longer so manifestly bad. Consider the realm of "badware" beyond viruses and worms. Most spyware, for example, purports to perform some useful function for the user, however halfheartedly it delivers. The nefarious Jessica Simpson screensaver does in fact show images of Jessica Simpson—and it also modifies the operation of other programs to redirect Web searches and installs spyware programs that cannot be uninstalled.⁹² The popular file-sharing program KaZaA, though advertised as "spyware-free," contains code that users likely do not want. It adds icons to the desktop, modifies Microsoft Internet Explorer, and installs a program that cannot be closed by clicking "Quit." Uninstalling the program does not unin-

stall all these extras along with it, and the average user does not have the know-how to get rid of the code itself. FunCade, a downloadable arcade program, automatically installs spyware, adware, and remote control software designed to turn the PC into a zombie when signaled from afar. The program is installed while Web surfing. It deceives the user by opening a pop-up ad that looks like a Windows warning notice, telling the user to beware. Click “cancel” and the download starts.⁹³

What makes such badware bad is often subjective rather than objective, having to do with the level of disclosure made to a consumer before he or she installs it. That means it is harder to intercept with automatic antivirus tools. For example, VNC is a free program designed to let people access other computers from afar—a VNC server is placed on the target machine, and a VNC client on the remote machine. Whether this is or is not malware depends entirely on the knowledge and intentions of the people on each end of a VNC connection. I have used VNC to access several of my own computers in the United States and United Kingdom simultaneously. I could also imagine someone installing VNC in under a minute after borrowing someone else’s computer to check e-mail, and then using it later to steal personal information or to take over the machine. A flaw in a recent version of VNC’s password processor allowed it to be accessed by anyone⁹⁴—as I discovered one day when my computer’s mouse started moving itself all over the screen and rapid-fire instructions appeared in the computer’s command window. I fought with an unseen enemy for control of my own mouse, finally unplugging the machine the way some Morris worm victims had done twenty years earlier. (After disconnecting the machine from the network, I followed best practices and reinstalled everything on the machine from scratch to ensure that it was no longer compromised.)

BEYOND BUGS: THE GENERATIVE DILEMMA

The burgeoning gray zone of software explains why the most common responses to the security problem cannot solve it. Many technologically savvy people think that bad code is simply a Microsoft Windows issue. They believe that the Windows OS and the Internet Explorer browser are particularly poorly designed, and that “better” counterparts (Linux and Mac OS, or the Firefox and Opera browsers) can help protect a user. This is not much added protection. Not only do these alternative OSes and browsers have their own vulnera-

bilities, but the fundamental problem is that the point of a PC—regardless of its OS—is that its users can easily reconfigure it to run new software from anywhere.

When users make poor decisions about what new software to run, the results can be devastating to their machines and, if they are connected to the Internet, to countless others' machines as well. To be sure, Microsoft Windows has been the target of malware infections for years, but this in part reflects Microsoft's dominant market share. Recall Willie Sutton's explanation for robbing banks: that's where the money is.⁹⁵ As more users switch to other platforms, those platforms will become more appealing targets. And the most enduring way to subvert them may be through the front door, asking a user's permission to add some new functionality that is actually a bad deal, rather than trying to steal in through the back, silently exploiting some particular OS flaw that allows new code to run without the user or her antivirus software noticing.

The Microsoft Security Response Center offers "10 Immutable Laws of Security."⁹⁶ The first assumes that the PC is operating exactly as it is meant to, with the user as the weak link in the chain: "If a bad guy can persuade you to run his program on your computer, it's not your computer anymore."⁹⁷ This boils down to an admonition to the user to be careful, to try to apply judgment in areas where the user is often at sea:

That's why it's important to never run, or even download, a program from an untrusted source—and by "source," I mean the person who wrote it, not the person who gave it to you. There's a nice analogy between running a program and eating a sandwich. If a stranger walked up to you and handed you a sandwich, would you eat it? Probably not. How about if your best friend gave you a sandwich? Maybe you would, maybe you wouldn't—it depends on whether she made it or found it lying in the street. Apply the same critical thought to a program that you would to a sandwich, and you'll usually be safe.⁹⁸

The analogy of software to sandwiches is not ideal. The ways in which we pick up code while surfing the Internet is more akin to accepting a few nibbles of food from hundreds of different people over the course of the day, some established vendors, some street peddlers. Further, we have certain evolutionary gifts that allow us to directly judge whether food has spoiled by its sight and smell. There is no parallel way for us to judge programming code which arrives as an opaque ".exe." A closer analogy would be if many people we encountered over the course of a day handed us pills to swallow and often conditioned en-

trance to certain places on our accepting them. In a world in which we routinely benefit from software produced by unknown authors, it is impractical to apply the “know your source” rule.

Worse, surfing the World Wide Web often entails accepting and running new code. The Web was designed to seamlessly integrate material from disparate sources: a single Web page can draw from hundreds of different sources on the fly, not only through hyperlinks that direct users to other locations on the Web, but through placeholders that incorporate data and code from elsewhere into the original page. These Web protocols have spawned the massive advertising industry that powers companies like Google. For example, if a user visits the home page of the *New York Times*, he or she will see banner ads and other spaces that are filled on the fly from third-party advertising aggregators like Google and DoubleClick. These ads are not hosted at nytimes.com—they are hosted elsewhere and rushed directly to the user’s browser as the nytimes.com page is rendered. To extend Microsoft’s sandwich metaphor: Web pages are like fast food hamburgers, where a single patty might contain the blended meat of hundreds of cows spanning four countries.⁹⁹ In the fast food context, one contaminated carcass is reported to be able to pollute eight tons of ground meat.¹⁰⁰ For the Web, a single advertisement contaminated with bad code can instantly be circulated to those browsing tens of thousands of mainstream Web sites operated entirely in good faith. To visit a Web site is not only to be asked to trust the Web site operator. It is also to trust every third party—such as an ad syndicator—whose content is automatically incorporated into the Web site owner’s pages, and every *fourth* party—such as an advertiser—who in turn provides content to that third party. Apart from advertising, generative technologies like RSS (“really simple syndication”) have facilitated the automated repackaging of information from one Web site to another, creating tightly coupled networks of data flows that can pass both the latest world news and the latest PC attacks in adjoining data packets.

Bad code through the back door of a bug exploit and the front door of a poor user choice can intersect. At the Black Hat Europe hacker convention in 2006, two computer scientists gave a presentation on Skype, the wildly popular PC Internet telephony software created by the same duo that invented the KaZaA file-sharing program.¹⁰¹ Skype is, like most proprietary software, a black box. It is not easy to know how it works or what it does except by watching it in action. Skype is installed on millions of computers, and so far works well if not flawlessly. It generates all sorts of network traffic, much of which is unidentifi-

able even to the user of the machine, and much of which happens even when Skype is not being used to place a call. How does one know that Skype is not doing something untoward, or that its next update might not contain a zombie-creating Trojan horse, placed by either its makers or someone who compromised the update server? The Black Hat presenters reverse engineered Skype enough to find a few flaws. What would happen if they were exploited? Their PowerPoint slide title may only slightly exaggerate: “Biggest Botnet Ever.”¹⁰² Skype is likely fine. I use it myself. Of course, I use VNC, too, and look where that ended up. The most salient feature of a PC is its openness to new functionality with minimal gatekeeping. This is also its greatest danger.

PC VS. INFORMATION APPLIANCE

PC users have increasingly found themselves the victims of bad code. In addition to overtly malicious programs like viruses and worms, their PCs are plagued with software that they have nominally asked for that creates pop-up windows, causes crashes, and damages useful applications. With increasing pressure from these experiences, consumers will be pushed in one of two unfortunate directions: toward independent information appliances that optimize a particular application and that naturally reject user or third-party modifications, or toward a form of PC lockdown that resembles the centralized control that IBM exerted over its rented mainframes in the 1960s, or that CompuServe and AOL exerted over their information services in the 1980s. In other words, consumers find themselves frustrated by PCs at a time when a variety of information appliances are arising as substitutes for the activities they value most. Digital video recorders, mobile phones, BlackBerries, and video game consoles will offer safer and more consistent experiences. Consumers will increasingly abandon the PC for these alternatives, or they will demand that the PC itself be applanicized.

That applanicization might come from the same firms that produced some of the most popular generative platforms. Microsoft’s business model for PC operating systems has remained unchanged from the founding days of DOS through the Windows of today: the company sells each copy of the operating system at a profit, usually to PC makers rather than to end users. The PC makers then bundle Windows on the machine before it arrives at the customer’s doorstep. As is typical for products that benefit from network externalities, having others write useful code associated with Windows, whether a new game,

business application, or utility, makes Windows more valuable. Microsoft's interest in selling Windows is more or less aligned with an interest in making the platform open to third-party development.

The business models of the new generation of Internet-enabled appliances are different. Microsoft's Xbox 360 is a video game console that has as much computing power as a PC.¹⁰³ It is networked, so users can play games against other players around the world, at least if they are using Xboxes, too. The business model differs from that of the PC: it is Gillette's "give them the razor, sell them the blades." Microsoft loses money on every Xbox it sells. It makes that money back through the sale of games and other software to run on the Xbox. Third-party developers can write Xbox games, but they must obtain a license from Microsoft before they can distribute them—a license that includes giving Microsoft a share of profits.¹⁰⁴ This reflects the model the video game console market has used since the 1970s. But the Xbox is not just a video game console. It can access the Internet and perform other PC-like functions. It is occupying many of the roles of the gamer PC without being generative. Microsoft retains a privileged position with respect to reprogramming the machine, even after it is in users' hands: all changes must be certified by Microsoft. While this action would be considered an antitrust violation if applied to a PC operating system that enjoyed overwhelming market share,¹⁰⁵ it is the norm when applied to video game consoles.

To the extent that consoles like the Xbox take on some of the functions of the PC, consumers will naturally find themselves choosing between the two. The PC will offer a wider range of software, thanks to its generativity, but the Xbox might look like a better deal in the absence of a solution to the problem of bad code. It is reasonable for a consumer to factor security and stability into such a choice, but it is a poor choice to have to make. As explained in Chapter Five, the drawbacks of migration to non-generative alternatives go beyond the factors driving individual users' decisions.

Next-generation video game consoles are not the only appliances vying for a chunk of the PC's domain. With a handful of exceptions, mobile phones are in the same category: they are smart, and many can access the Internet, but the access is channeled through browsers provided and controlled by the phone service vendor. The vendor can determine what bookmarks to preinstall or update, what sites to allow or disallow, and, more generally, what additional software, if any, can run on the phone.¹⁰⁶ Many personal digital assistants come with software provided through special arrangements between device and software vendors, as Sony's Mylo does with Skype. Software makers with-

out deals cannot have their code run on the devices, even if the user desires it. In 2006, AMD introduced the “Telmex Internet Box,” which looks just like a PC but cannot run any new software without AMD’s permission. It will run any software AMD chooses to install on it, even after the unit has been purchased.¹⁰⁷ Devices like these may be safer to use, and they may seem capacious in features so long as they offer a simple Web browser, but by limiting the damage that users can do through their own ignorance or carelessness, the appliance also limits the beneficial activities that users can create or receive from others—activities they may not even realize are important to them when they are purchasing the device.

Problems with generative PC platforms can thus propel people away from PCs and toward information appliances controlled by their makers. Eliminate the PC from many dens or living rooms, and we eliminate the test bed and distribution point of new, useful software from any corner of the globe. We also eliminate the safety valve that keeps those information appliances honest. If TiVo makes a digital video recorder that has too many limits on what people can do with the video they record, people will discover DVR software like MythTV that records and plays TV shows on their PCs.¹⁰⁸ If mobile phones are too expensive, people will use Skype. But people do not buy PCs as insurance policies against appliances that limit their freedoms, even though PCs serve exactly this vital function. People buy them to perform certain tasks at the moment of acquisition. If PCs cannot reliably perform these tasks, most consumers will not see their merit, and the safety valve will be lost. If the PC ceases to be at the center of the information technology ecosystem, the most restrictive aspects of information appliances will come to the fore.

PC AS INFORMATION APPLIANCE

PCs need not entirely disappear as people buy information appliances in their stead. They can themselves be made less generative. Recall the fundamental difference between a PC and an information appliance: the PC can run code from anywhere, written by anyone, while the information appliance remains tethered to its maker’s desires, offering a more consistent and focused user experience at the expense of flexibility and innovation. Users tired of making the wrong choices about installing code on their PCs might choose to let someone else decide what code should be run. Firewalls can protect against some bad code, but they also complicate the installation of new good code.¹⁰⁹ As anti-virus, antispyware, and antibadware barriers proliferate, they create new chal-

lenges to the deployment of new good code from unprivileged sources. And in order to guarantee effectiveness, these barriers are becoming increasingly paternalistic, refusing to allow users easily to overrule them. Especially in environments where the user of the PC does not own it—offices, schools, libraries, and cyber cafés—barriers are being put in place to prevent the running of any code not specifically approved by the relevant gatekeeper.

Short of completely banning unfamiliar software, code might be divided into first- and second-class status, with second-class, unapproved software allowed to perform only certain minimal tasks on the machine, operating within a digital sandbox. This technical solution is safer than the status quo but, in a now-familiar tradeoff, noticeably limiting. Skype works best when it can also be used to transfer users' files, which means it needs access to those files. Worse, such boundaries would have to be built into the operating system—placing the operating system developer or installer in the position of deciding what software will and will not run. If the user is allowed to make exceptions, the user can and will make the *wrong* exceptions, and the security restrictions will too often serve only to limit the deployment of legitimate software that has not been approved by the right gatekeepers. The PC will have become an information appliance, not easily reconfigured or extended by its users.

* * *

The Internet Engineering Task Force's RFC 1135 on the Morris worm closed with a section titled "Security Considerations." This section is the place in a standards document for a digital environmental impact statement—a survey of possible security problems that could arise from deployment of the standard. RFC 1135's security considerations section was one sentence: "If security considerations had not been so widely ignored in the Internet, this memo would not have been possible."¹¹⁰

What does that sentence mean? One reading is straightforward: if people had patched their systems and chosen good passwords, Morris's worm would not have been able to propagate, and there would have been no need to write the memo. Another is more profound: if the Internet had been designed with security as its centerpiece, it would never have achieved the kind of success it was enjoying, even as early as 1988. The basic assumption of Internet protocol design and implementation was that people would be reasonable; to assume otherwise runs the risk of hobbling it in just the way the proprietary networks were hobbled. The cybersecurity problem defies easy solution, because any of the most obvious solutions to it will cauterize the essence of the Internet and

the generative PC.¹¹¹ That is the generative dilemma. The next chapter explains more systematically the benefits of generativity, and Chapter Five explores what the digital ecosystem will look like should our devices become more thoroughly applanized. The vision is not a pleasant one, even though it may come about naturally through market demand. The key to avoiding such a future is to give that market a reason not to abandon or lock down the PCs that have served it so well—also giving most governments reason to refrain from major intervention into Internet architecture. The solutions to the generative dilemma will rest on social and legal innovation as much as on technical innovation, and the best guideposts can be found in other generative successes in those arenas. Those successes have faced similar challenges resulting from too much openness, and many have overcome them without abandoning generativity through solutions that inventively combine technical and social elements.