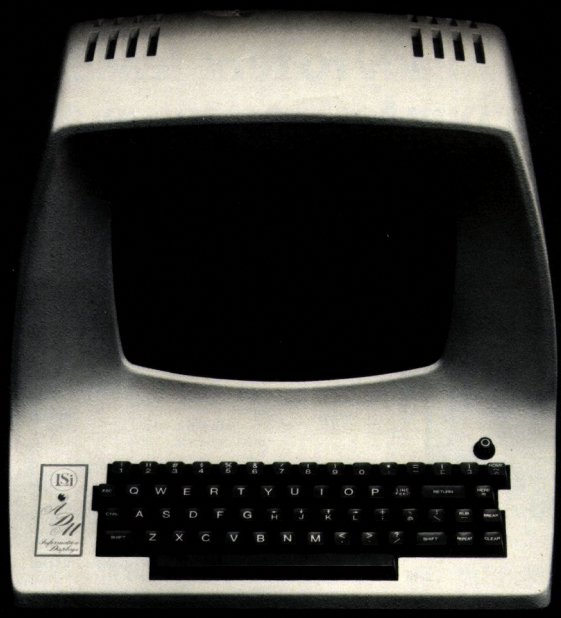
The Morris worm was released in November of 1988. It was launched surreptitiously from an MIT computer by graduate student Robert Tappan Morris at Cornell University, and spread to internet-connected computers running the BSD variant of UNIX. The worm was designed to be undetectable, but a design flaw led it to create far more copies of itself than Morris estimated, and resulted in the drastic over-taxing of all the computers on which it was installed. This in turn allowed for its immediate detection and the repair of the flaws that it exploited.

THE MORRIS WORM

BY CHRISTOPHER M. KELTY



THE MORRIS WORM was not a destructive worm, it only caused computers to slow and buckle under the weight of unnecessary processing. Nor was the intent of Morris clear: some speculate that the release was either premature or accidental (Spafford 1989; Eisenberg et. al. 1989). Nonetheless the event precipitated two different responses that have since become the focus of much attention and concern over the intervening years. Exploring these two responses reveals something about what "systemic risk" might or might not mean in the context of the Internet and how it relates to other uses of the concept.

The first and most direct response was that Morris became the first individual to be tried under the new Computer Fraud and Abuse Act of 1986, 18 U.S. Code Section 1030(a)(5)(A). Morris was tried, convicted and sentenced to three years of probation, 400 hours of community service, a fine of $10,050, and the costs of his supervision. The case was appealed, and the conviction upheld.

The second response was the creation by Defense Advanced Research Projects Agency (DARPA) of the Computer Emergency Response Team (CERT), in order to coordinate information and responses to computer vulnerabilities and security.

OF THE FIRST RESPONSE—the criminal prosecution—a couple of things stand out. The first is the continuing moral ambivalence about Morris' intentions. On the one hand, what Morris did, objectively, was to force certain security vulnerabilities to be fixed by writing a program that publicly exploited them. As the author of one official investigation, Eugene Spafford, pointed out, the code contained no commands that would harm a computer on which it ran, only commands intended to exploit vulnerabilities that allowed the code to copy itself and spread. On the other hand, his conviction for Fraud and Abuse clearly sends a different message—that this was a criminal act, and as the law had it, one that threatened not just citizens, but the federal government itself.

The practice of publicly demonstrating exploitable vulnerabilities in order to force vendors and system administrators to fix their systems has become established in the academic field of computer science, though it has been largely restricted to the publication of papers that demonstrate how to do it, rather than the release of software that actually does so. This puts Morris, who is now employed at MIT's AI lab and a successful researcher, squarely in the camp of what is known colloquially as "white hat hackers"—hackers, including both academic and corporate employees, who (demonstrate how to) exploit vulnerabilities in order to *make publicly visible* security flaws that need fixing. Morris' worm, from this standpoint looks more like incompetence as a white hat hacker, than the criminal action of a black hat hacker.[2] The moral ambivalence mirrors that around many of the hacker-cum-Silicon Valley success stories that might be cited in this instance.

In terms of the criminality of the Morris worm, one might ask: is the risk that such actions pose a *systemic* risk? The Morris Worm was neither designed to, nor did it cause harm of a particular sort (theft of documents or information, deletion or destructive interference, spam, porn, terrorist intervention, etc.). Rather, its effect was more general in that it caused the Internet, as a collection of interconnected computers, to do something it was not designed to do, and as a result, slow down or cease to function properly.

But is such an effect an issue of "systemic risk?" In part the answer may rest on what is defined as the system, and in particular whether the *system* or the *risk* is perceived as the "emergent" property (i.e. something which emerges through the interaction of parts, but cannot be reduced to, or predicted by those interactions). In the case of the Internet, the system itself is the emergent part: what we call the Internet is only the effect of tens of millions of devices all operating with standardized software-based protocols that govern how they connect, transfer information and disconnect. The platform that emerges is flexible, reconfigurable, asynchronous and without any pre-designed structure to it. Worms and viruses affect this emergent system by affecting software on many particular computers; in the case of the Morris worm, by changing the function of the email management software called `sendmail`, and the name lookup service called `finger`. The particular vulnerabilities that allow a worm or virus to do this (such as "buffer overflow" vulnerabilities) are the proper quarry of the computer security researcher.

The terminology of *worms* and *viruses* express different conceptions of propagation in terms of computer programs. Viruses operate on the analogy of biological viruses by inserting a bit of code into a running program which exploits a vulnerability to allow it to replicate, and potentially, to do harm. A worm by contrast (shortened from tapeworm) is a complete program more like a parasite or symbiont; it reproduces itself via the very vulnerabilities that it exploits. In both cases, individual (networked) computers are the locus and necessary starting point: because networked operating system software is designed to create myriad forms of connections with other computers, and hence bring a network into being, it can be exploited to spread worms and viruses similar to how infections or rumors spread. There is no difference, therefore, between the risk to all of the infected computers combined, and the risk to the "system" understood as the network that emerges out of the interconnection of machines. Or to put it differently, security researchers understand the nature of the risk of a virus or worm to be simply the risk posed by a virus to a particular computer multiplied by the number of computers that are affected. In this sense, worms or viruses do not pose a *systemic* risk—i.e. a new risk that emerges out of or on top of aggregate risks to known components. Rather they are of a piece with the emergence of the network itself.

Contrast this with systemic risk in a case like catastrophe insurance. In the case of catastrophe insurance it is not necessarily the system that is emergent, but the risk. Individual policies have well-defined risks, but the risk of a portfolio of policies cannot be predicted by the aggregate risk of all the individual policies. Similarly, there is nothing analogous to the Internet as an "emergent network" that results from the issuing of policies—though catastrophe insurance policies can clearly be said to interact at some level (especially when events occur). As such there is a "systemic risk" at play that is different in kind, not only degree, from the risk that pertains to individual policies. The comparison is uneven, however, since the case of insurance already builds on the concept of population-level risk, and not just that of individuals—there is no obvious point of comparison here to the relationship between individual computers and a network.

Nonetheless, the idea that computer security risks are not "emergent" explains the often indignant response of computer security researchers (and white hat hackers) to vulnerabilities: there is no mystery, there is no "emergent risk," the vulnerabilities can and should be fixed by users and vendors by fixing all the individual software components.

There are other uses of "systemic" in the context of computer security. The language of viruses and worms clearly designates a threat to a system understood as an organism. The language of *payloads* (often also used in the biological field) designates the specific threat or violence a virus or worm might visit on a computer and its associated network. At least one paper uses the language of an *ecology* of "security and policy flaws" in which it is possible for worms

1 Morris' reputation has suffered little. For many hackers (Kevin Mitnick being the most famous), a conviction is hardly a setback, and in many cases provides a boost to their reputations. Morris is today also a partner in the alternative venture capital firm Y Combinator, where his bio delightfully reads "In 1988 his discovery of buffer overflow first brought the Internet to the attention of the general public."

and viruses to thrive (Weaver et. al. 2003:16): application design, buffer overflows, privileges, application deployed widely, economic costs of development and testing of software, patch deployment, and "monocultures" (i.e. the absence of a diversity of machines and software).

For the most part, the Morris worm—as a problem of hackers, criminals, computer security researchers and software vendors—is concerned with the possibility of understanding, making visible and controlling vulnerabilities in the parts, in order to safeguard an emergent system that forms through their interaction. Risk is almost always equated to vulnerabilities (known and unknown) in this sense.

BY CONTRAST, the other response, the formation of the Computer Emergency Response Team, ties the event of the Morris worm directly to the rise of thinking about critical infrastructure protection. The DARPA funded project, which is headquartered at Carnegie Mellon's Software Engineering Institute, publishes a variety of announcements, analyses, fixes and warnings concerning software and networking vulnerabilities. In 1997 they wrote a report (Ellis et. al. 1997) to the Presidents Commission on Critical Infrastructure Protection that reported a list of problems that CERT has continually encountered (similar to the ecology cited earlier, but not so-named). A handful of publications and presentations that identify problems of interdependency have been published, and the organization has focused some of its energy on creating tools, mechanisms, textbooks and guidelines for "software assurance", as well as topics like "resiliency engineering management" and "vulnerability disclosure."

The rise of "Critical Infrastructure Interdependency" research that began with the publication of Rinaldi et.al. (2001) has grown out of and alongside these institutions. The notion of "infrastructure interdependency" is a more apt conceptual analog to "systemic risk" than are worms and viruses (which in this context look more like specific techniques, rather than events in themselves). CII research suggests that it is not the system that is emergent, but the risk. Individual systems (or infrastructures) may be emergent, as the Internet is, and even highly designed systems such as the electrical power grid might also exhibit emergent features (Watts and Strogatz 1998). However, it is the interaction between systems or infrastructures that introduces risks that cannot be predicted or reduced to the component parts. CII researchers are therefore intensely concerned with the concrete points of connection between systems, a much discussed example of which is supervisory control and data acquisition (SCADA) software, especially when it is deployed in the context of the Internet as a tool for managing industrial processes. The "emergent" risk comes from the unpredictable vulnerabilities that obtain when, for instance, electrical power grids are monitored by and dependent upon networked computers. The language of "cascades" and contagious risk appears in this research with as much regularity as it does in the domain of finance (e.g. the bank contagion literature).

The Morris worm reveals something about what "systemic risk" might or might not mean in the context of the Internet.

For its part, the academic field of computer security research seems to remain at a distance from the related concerns of systemic risk in finance, public health, or defense, despite being well-funded by the likes of DARPA and well-represented by agencies like the National Security Agency (whose National Computer Security Center was for a long period headed by none other than Robert Morris Sr.). For most computer security researchers, flaws and vulnerabilities are confined to the operating system level of networked computers, or at most, extend to social or policy vulnerabilities, such as physical access to computers, poor management of accounts and privileges, or non-existent monitoring of personal computers connected to the Internet. They have historically not been much concerned with the interdependency of two networked systems, like the electrical grid and the Internet. A certain hubris, perhaps, follows from the separation of worms/viruses and their "payload"—and the field is alive with imaginaries about exploitation by evil-doers, rather than collapse or break-down due to normal design or usage. The idea of a "normal accident" is almost completely foreign to existing computer security research. Intent, even in the attenuated form that resulted in Morris' conviction, is an obsession of this field. By contrast, breakdowns and the failure of software to function as expected has largely been the subject of software engineering, which has been berating itself for a very long time now (40 years, beginning with the famous 1968 NATO report on Software Engineering). Instead of security, such concerns are largely the domain of software development practices, quality assurance systems, and an ever-increasing number of standards intended to manage the problem of poorly designed, poorly performing software.

By way of conclusion, an interesting point of convergence has emerged just in the past year. The recent W32.Stuxnet worm targeted SCADA software made by Siemens used to conduct process engineering and control the operation of large-scale plants, like Iran's new nuclear power plant. It's not at all clear at this point that this should be called an *Internet* worm or virus, because it could not have infected the computers it infected without someone physically inserting a USB stick with the Stuxnet code into a computer running the Siemens SCADA software. The worm replicated itself through an internally networked system and it apparently opens up control to outsiders, but it does not go on to affect the function of the Internet in any way, only the operation of the plant in question. Here, again, the risk is "emergent" but only in the sense that the longstanding attempt to create computer controlled industrial processes has itself produced unpredictable vulnerabilities. At a technical level such vulnerabilities are not different in kind than those the Morris worm exploited: they result from poor engineering, overlooked weaknesses, or poor security practice. But at a more "systemic" level they are different in kind since they affect not only the operation of the computers themselves, but the physical operation of a plant or "interdependent" system. ❑

CHRISTOPHER KELTY *is Associate Professor at the Center for Society and Genetics and the Department of Information Studies, UCLA.*