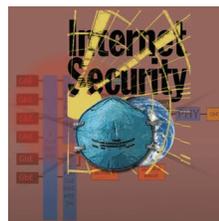


# Worm Epidemics in High-Speed Networks



**Future worm epidemics might spread at unprecedented rates in high-speed networks. A comprehensive automated defense system will be the only way to contain new threats but could be too risky to implement without more reliable detection accuracy and better real-time traffic analysis.**

Thomas M.  
Chen

Southern Methodist  
University

Jean-Marc  
Robert

Alcatel Canada

Since the Melissa macro virus struck Microsoft Windows users in late March 1999, viruses and worms have become a common and persistent problem for all computer users. For various practical reasons, many machines remain unprotected by up-to-date software patches or antivirus software, and the emergence of the Internet has made it easy to shut down a sizable number of vulnerable systems either directly through a denial-of-service attack, or indirectly through network congestion.

In its eighth annual survey of computer crime in the United States, the Computer Security Institute ([www.gocsi.com](http://www.gocsi.com)), in collaboration with the San Francisco Federal Bureau of Investigation's Computer Intrusion Squad, reported that malicious software impacted 82 percent of surveyed organizations, causing an average loss of \$200,000. Computer Economics ([www.computereconomics.com](http://www.computereconomics.com)), an IT research firm, estimates the annual global impact of viruses and worms to be in the billions of dollars.

In particular, worms have become more prevalent as Internet connectivity, including always-on broadband access, has become ubiquitous. Unlike viruses, which attach parasitically to a normal program, worms are stand-alone automated programs designed to seek out and infect vulnerable computers with a copy of themselves. They are thus intrinsically dependent on a network and, as the "Famous Computer Worms" sidebar describes, have caused problems since the early days of the Arpanet, the forerunner of the Internet.

Ironically, emerging high-speed networks will likely accelerate the spread of worms, especially those like Code Red and SQL Slammer that are mostly limited by available bandwidth. As network rates increase, the time available to respond to worm epidemics may shorten to seconds before the entire vulnerable population is saturated. Ad hoc manual defenses will be much too slow; only an automated defense system might be capable of detecting and isolating a new worm so quickly. Unfortunately, although this idea has been around for years, many long-standing technical problems require better solutions.

## HOW A COMPUTER WORM SPREADS

A computer worm that randomly scans new hosts to infect can be expected to follow the simple epidemic model known from biological epidemiology.<sup>1</sup> This model assumes that a population of constant  $N$  hosts are initially all vulnerable but uninfected except for a small number that are infected and contagious. These *susceptibles* and *infectives*, respectively, mix randomly, with an infection parameter  $\beta$  characterizing the rate of infection between susceptible-infective pairs.

Once infected, a host remains permanently infected; the model does not allow for recovery or deaths during the epidemic's timescale. More realistically, certain hosts might be invulnerable to infection, but those cases are simply discounted from the population of interest. Expressed mathematically, if  $I_t$  is the number of infectives in the population at time  $t$ , then the simple epidemic follows the logistic curve

## Famous Computer Worms

Distributed processing pioneers at the Xerox Palo Alto Research Center—site of the first Ethernet connection—coined the term *worm* in 1979 from the autonomous, data-deleting programs called “tapeworms” in John Brunner’s science fiction novel, *The Shockwave Rider* (Del Ray, 1975). Since then, a number of worms have caused havoc on the Internet.

The Morris worm was the first to spread “in the wild.” Robert T. Morris Jr., a Cornell University graduate student, launched what was intended to be a benign experiment via remote login from the MIT Artificial Intelligence Laboratory on 2 November 1988. A bug in the program caused it to replicate much faster than anticipated, and by the next day it had crashed 10 percent of the fledgling Internet. Morris’s creation infected around 6,000 Unix machines nationwide, causing up to \$100 million in damage, and led directly to the creation of the Computer Emergency Response Team Coordination Center ([www.cert.org](http://www.cert.org)).

The Melissa worm first appeared on 26 March 1999 and targeted Microsoft Word and Outlook users. The creator, David L. Smith, named the worm after an exotic dancer and distributed it in a Usenet discussion group as a Word file listing passwords to pornographic Web sites. When a user downloaded and opened the infected file, it sent itself to the first 50 names in the user’s Outlook address book. Although Melissa was intended as a joke—the payload consisted of quotations from the animated TV show *The Simpsons*—it infected around 100,000 computers in the first weekend, congesting e-mail servers around the world.

On 4 May 2000, Onel de Guzman, a college dropout in the Philippines, unleashed the virulent Lovebug worm. Sent as an e-mail Visual Basic script attachment titled “ILOVEYOU,” it spawned copies of itself to everyone in the victim’s Outlook address book. The worm infected tens of millions of computers worldwide, shutting down e-mail servers and causing billions of dollars in damage to businesses.

In July 2001, two major worm epidemics attracted media attention. The Code Red worm exploited a security hole in Microsoft’s Internet Information Server software. The first version spread slowly, but a more virulent offshoot infected more than 350,000 systems running IIS in less than a day. After a period of hibernation, the malicious worm caused the compromised servers to flood the White House Web site with garbage data and defaced Web pages with the message “Hacked by Chinese!” Only a flaw in the program prevented Code Red from realizing its destructive potential.

The prolific Sircam worm, which appeared in the same month, spread primarily as an e-mail attachment with a randomly chosen subject line. When a user opened the infected attachment, it randomly selected a file from the user’s My

Documents folder, infected it, and sent it to e-mail addresses in the computer’s Outlook address book or Internet cache. The worm was programmed to delete all data files on infected hard drives on a certain day three months later, but an error in the code prevented the attack from occurring.

On 24 January 2003, the SQL Slammer worm began writhing its way through the Internet. Although the worm, which exploited vulnerabilities in Microsoft SQL Server, did not disturb any Web pages or harm any files on infected machines, it spread at an alarming rate, snarling global Internet traffic and disrupting corporate networks, before it fizzled out.

Two major worm epidemics appeared within a week in 2003. Discovered August 11, the Blaster worm infected half a million computers during the summer by exploiting a distributed component object model remote procedure call vulnerability on Windows XP and Windows 2000 PCs. A teenager created the most famous variant, dubbed Lovesan, which unsuccessfully launched a denial-of-service (DoS) attack against the Microsoft Windows Update Web server and caused operating systems on some users’ machines to reboot or become unresponsive.

On 18 August, the Sobig.f worm surfaced, spreading rapidly among thousands of Windows PCs by e-mail. Like its earlier incarnations, which appeared serially beginning in January, it exploited open proxy servers to turn infected machines into spam engines. At its peak, Sobig.f reportedly accounted for one in every 17 messages and produced more than 1 million copies of itself within the first 24 hours. The Sobig worm variants “spoofed” valid e-mail messages, primarily the Microsoft home address, to disguise their malicious purposes.

Mydoom, a mass-mailing e-mail worm that emerged on 26 January 2004, followed the growing trend of worms installing a backdoor in infected computers, thereby enabling hackers to gain remote access to data such as passwords and credit card numbers. Designed to launch a timed DoS attack against Microsoft and the SCO Group, Mydoom replicated up to 1,000 times a minute and reportedly flooded the Internet with 100 million infected messages in its first 36 hours.

On 30 April 2004, a new worm began circulating on the Internet that exploited a Local Security Authority Subsystem Service vulnerability in Microsoft Windows 2000, Windows Server 2003, and Windows XP systems. Timed to strike over the weekend after security personnel had gone home, Sasser infected more than a million PCs worldwide within a few days, causing them to repeatedly shut down and reboot. The worm spread automatically by scanning random IP addresses for vulnerable systems, especially residential computers with always-on broadband connections.

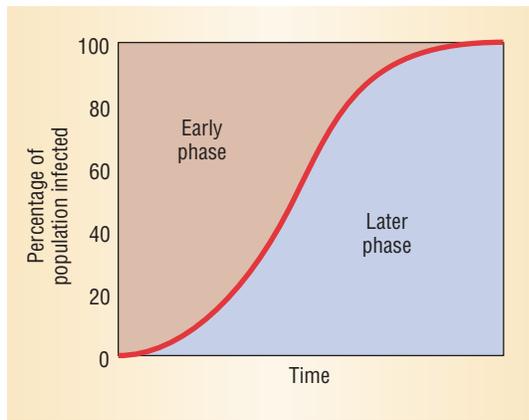
$$I_t = \frac{I_0 N}{I_0 + (N - I_0)e^{-\beta N t}},$$

which has the familiar S-shape shown in Figure 1.

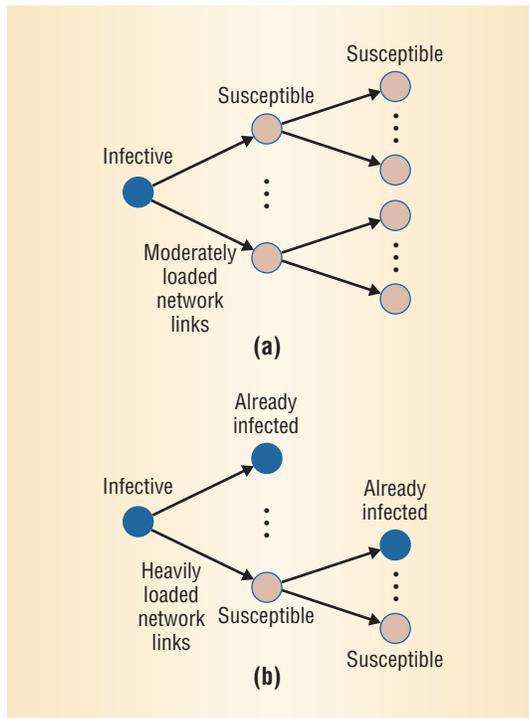
Under these assumptions, the entire susceptible population eventually becomes infected at a rate dependent on  $\beta$ —the larger the infection param-

eter, the faster the infection. As Figure 1 shows, a simple epidemic goes through two distinct phases. In the early phase, the number of infectives is a small fraction of the population, and the growth is approximately exponential according to  $I_t = I_0 e^{\beta N t}$ . As infectives saturate the population, the rate of spreading slows down in the later phase because randomly scanned targets are more likely to be

**Figure 1. Simple epidemic logistic curve. Growth is exponential in the early phase, slowing down in the later phase as infectives saturate the population.**



**Figure 2. Random-scanning worm epidemic. (a) In the early phase, an infected host scans likely susceptibles, which in turn scan other susceptibles, leading to exponential growth. (b) In the later phase, the epidemic slows down due to inefficient scanning and network congestion.**



infected already, and few susceptibles remain to come into contact with infectives.

Figure 2 illustrates these two phases for a random-scanning worm. In the early phase, an infected host scans a number of likely susceptibles, which in turn scan other hosts, leading to exponential growth. The random scanning is relatively efficient in the early phase because a high percentage of targets are likely to be susceptibles. Also, if the population is large, scans are not likely to overlap much—that is, a host will not receive multiple hits.

The volume of scanning traffic increases with the number of infected hosts, resulting in network congestion similar to a denial-of-service (DoS) attack. Side effects of the worm traffic, such as Internet Control Message Protocol “destination/port unreachable” messages returned for unsuccessful scans, can aggravate this problem. Network congestion is manifested by long packet delays and high packet loss, which serve to dampen a worm

outbreak because infected hosts cannot easily reach other hosts. Active defensive measures such as packet filtering by routers will also help curtail the epidemic.

The population size  $N$  can be viewed as the entire  $2^{32}$  IP address space in the worst case. For a given  $N$  and time  $t$ , the critical factor in the spreading rate is  $\beta(1 - I_t/N)$ , which represents the average number of secondary infections by each worm per unit of time.

## EXAMPLES OF FAST WORMS

Examining two worms, Code Red and SQL Slammer, illustrates how random-scanning worms spread so quickly on the Internet.

### Code Red

The Code Red worm achieved its rapid infection rate through parallelism. At least three versions of the worm attempted to exploit a buffer overflow in Microsoft’s Internet Information Server software, which the company revealed on 18 June 2001. The flaw resided in a component used to support indexing and thus speed up searches. The Indexing Service ISAPI filter in IIS did not check the length of data in incoming HTTP GET request messages, enabling a carefully crafted packet to cause a buffer overflow. By exploiting this hole, a hacker could execute arbitrary code and gain full system-level access to the target server.

The first version of Code Red appeared about a month later on 12 July 2001.<sup>2</sup> CRv1 scanned the Internet for vulnerable servers, using Transmission Control Protocol port 80 as its attack vector. To compensate for the inherent latency in setting up a TCP connection with potential targets, the worm employed multiple threads. Upon infecting a machine, CRv1 set itself up in memory and generated up to 100 new threads, each an exact replica of the original worm. Thus, the propagation rate depended on an infected machine’s multitasking capability and how many threads it could block.

CRv1 spread slowly because a programming error caused it to generate identical, rather than random, lists of IP addresses on each infected host. On 19 July, a second version of Code Red appeared with the error apparently fixed. CRv2 spread much faster, infecting more than 359,000 machines within 14 hours. At its peak, the worm infected 2,000 hosts every minute.

On 4 August, a new worm self-named Code Red II began exploiting the same security hole in IIS Web servers.<sup>3</sup> After infecting a host, it lay dormant for one to two days and then rebooted the machine.

After rebooting, the worm activated 300 threads to probe other machines. About one out of eight IP addresses that CRII generated were completely random, half were within the same class A range of the infected host's address, and three out of eight addresses were within the same class B range of the infected host's address. The enormous number of parallel threads thereby created a flood of scans, compromising about 400,000 systems and causing considerable network congestion.

### SQL Slammer

Like Code Red, the SQL Slammer worm that raced through the Internet in late January 2003 exploited a buffer overflow vulnerability—in this case in Microsoft SQL Server 2000 and its free redistributable version, MSDE 2000—announced by the company six months earlier.<sup>4</sup> This worm, also known as Sapphire and Helkern, achieved a record-breaking infection rate through its surprising simplicity. Much smaller than the 4-Kbyte Code Red and other previous worms, it fit in the 376-byte payload of one User Datagram Protocol packet. A single UDP packet directed to port 1434, the default port for the database software's resolution service, was sufficient to cause a buffer overflow in the service and install a copy of the worm.

The absence of a payload suggests that SQL Slammer's sole purpose was propagation. The spreading rate was reportedly fast enough to infect 90 percent of vulnerable hosts, around 75,000 servers, within 10 minutes.<sup>5</sup> In the first minute, the infection doubled every 8.5 seconds, and it hit a peak scanning rate of 55 million scans per second after only three minutes. In contrast, the Code Red infection doubled in 37 minutes but infected more machines.

SQL Slammer spread rapidly by causing infected computers to generate UDP packets carrying the worm at the maximum rate of the machine or network link—up to 26,000 probes per second, with an average rate per machine of 4,000 probes per second.<sup>5</sup> This approach thus avoided the delays and overhead in the form of control messages associated with setting up a TCP connection.

### WORMS AND HIGH-SPEED NETWORKS

A worm similar to Code Red or SQL Slammer could likely achieve a far higher infection rate and saturate the target population much more quickly in a high-speed network. Such networks can increase  $\beta(1 - I_i/N)$  by making it easier for infected hosts to communicate with potential targets. The simple epidemic formula can be rearranged as

$$T_p = \frac{\ln P(N - I_0) - \ln(1 - P)I_0}{\beta N},$$

where  $T_p$  represents the time it takes to infect a fraction  $P$  of the population—that is, to infect  $PN$  hosts. This result implies that if a worm finds the bandwidth to double its probe rate, effectively doubling the infection parameter  $\beta$ , it could saturate the target population in half the time.

Worms able to avoid the inefficient scanning and network congestion that slow down simple epidemics in the later phase could spread even faster than random-scanning worms such as Code Red and SQL Slammer, reducing the available response time to a matter of seconds in high-speed networks.<sup>6</sup> Such worms could, for example, compile a list of potential hosts during a preliminary reconnaissance phase to avoid wasteful probes of invulnerable targets. They could also minimize duplication of effort by coordinating the probing activities of all replicas through a Web site or Internet relay chat channel.

### AUTOMATIC WORM DETECTION

Researchers have long recognized the need for automatic detection and containment of new worms.<sup>7</sup> Traditional defenses against malware consist of an ad hoc combination of antivirus software, operating system patches, and network security equipment such as firewalls. However, many users are unwilling to expend the effort and endure the inconvenience of frequently updating software and applying patches. In addition, while firewalls, routers, intrusion detection systems, and other network security equipment are useful for limited protection of enterprise networks, they are not currently designed to work cooperatively in a distributed, automated defense system.

### Intrusion detection systems

The idea of an automated intrusion detection system (IDS) can be traced as far back as 1980, when James Anderson proposed using statistical analysis to recognize unusual behavior in computer systems.<sup>8</sup> The US Navy sponsored an early prototype called the Intrusion Detection Expert System in the mid-1980s,<sup>9</sup> and commercial IDS products began appearing in the 1990s.

An IDS performs three basic functions: It collects raw data from sensors that monitor and record activities in the hosts or network, analyzes that data to classify activities as normal or suspicious, and triggers a response to any suspicious activity it con-

**A worm could achieve a higher infection rate and saturate the target population more quickly in a high-speed network.**

**The central problem in any IDS is accurately analyzing and classifying monitored activities.**

siders sufficiently serious. A response usually is simply an alarm that the IDS sends to the network administrator for further action or diagnosis.

In 1991, IBM proposed a *digital immune system* that combined intrusion detection with a more active response.<sup>10</sup> The approach, which Symantec has incorporated into its commercial antivirus products, was inspired by the human immune system's response to a viral infection.<sup>11</sup> A digital immune system is designed to automatically detect new worms and viruses, report them to a central analysis center, automatically create new signatures, and coordinate dissemination of updated signatures. All parts of the system must work properly for it to be effective, which in practice is difficult to guarantee.

### Detection accuracy

The central problem in any IDS is accurately analyzing and classifying monitored activities. In the digital immune system, the problem is discriminating “self” from “nonself.” In biological terms, self refers to any cells belonging to the host body, while nonself is foreign objects such as pathogens or parasites.<sup>12</sup> An ideal intrusion detection system avoids both false positives (unnecessary alarms) and false negatives (missed intrusions), but current technology is not close to attaining perfect accuracy or reliability.

The two basic IDS approaches to data analysis are misuse detection and anomaly detection. In practice, most systems are based on misuse detection and augmented with anomaly detection.

**Misuse detection.** Commonly used in commercial IDS offerings, misuse detection defines a set of attack signatures and looks for matching behavior. This approach inherently depends on signature accuracy: If the signatures are too narrowly defined, some attacks might not be detected, resulting in false negatives; if signatures are too broadly defined, some benign behavior might cause false positives. Another critical limitation of signature-based intrusion detection is the inability to detect new worms that do not match a known signature and might attack an unknown or unannounced vulnerability.

**Anomaly detection.** In contrast, anomaly detection defines a statistical pattern for “normal” behavior and interprets any deviations from that pattern as suspicious. Although this approach can detect new attacks without a known signature, accurately defining normal behavior is problematic. In addition, only a small fraction of suspicious cases may truly be malicious—if every suspicious case raised

an alarm, behavior-based intrusion detection could result in a high rate of false positives.

Another problem with anomaly detection is the difficulty of identifying “wormlike” behavior. Worms can exhibit certain signs—a dramatic increase in network traffic volume, a steady increase in scans and probes, a sudden change in the traffic behavior of hosts—but these do not necessarily indicate a worm attack. For example, port scans are a normal part of the Internet's background traffic and can also contribute to sudden congestion.

### Real-time detection

Performing worm detection in real time is critical given the short window of time available for containing a fast worm epidemic. Modern firewalls and routers have this built-in capability, but actual worm traffic might constitute a minute fraction of the vast amounts of data that high-speed networks carry. In short, detecting rare events could require enormous processing power.

A fast worm epidemic also requires a close tie between real-time detection and an active response system, such as automatic reconfiguration of routers or firewalls, to block worm traffic. Typically, network administrators must sift through voluminous logs of data to identify real intrusions. This process would be much too slow and time-consuming for worm epidemics.

**T**he rate at which a worm epidemic spreads determines how many computing systems it can potentially infect and thus, ultimately, the disruption and cleanup costs. Only a comprehensive automated defense system will be able to quickly contain future worm outbreaks in high-speed networks. However, an automated response to a false alarm could trigger the wrong course of action. Moreover, whether the alarm is false or correct, the response must not unduly obstruct legitimate traffic.

Although researchers continue to work on improving worm detection accuracy and real-time traffic analysis, a practical solution thus far remains elusive. One possible alternative is to try to prevent a worm from spreading rather than react to an existing epidemic. Because infected hosts typically scan different IP addresses at very high rates, a logical preventive step would be to limit the rate of such scanning. This approach, combined with ingress filtering to prevent source-address “spoofing,” could sharply reduce malicious traffic in high-speed networks.

## References

1. D.J. Daley and J. Gani, *Epidemic Modelling: An Introduction*, Cambridge Univ. Press, 1999.
2. CERT Incident Note IN-2001-08, “Code Red’ Worm Exploiting Buffer Overflow in IIS Indexing Service DLL,” 19 July 2001; [www.cert.org/incident\\_notes/IN-2001-08.html](http://www.cert.org/incident_notes/IN-2001-08.html).
3. CERT Incident Note IN-2001-09, “Code Red II: Another Worm Exploiting Buffer Overflow in IIS Indexing Service DLL,” 6 Aug. 2001; [www.cert.org/incident\\_notes/IN-2001-09.html](http://www.cert.org/incident_notes/IN-2001-09.html).
4. CERT Advisory CA-2003-04, “MS-SQL Server Worm,” 27 Jan. 2003; [www.cert.org/advisories/CA-2003-04.html](http://www.cert.org/advisories/CA-2003-04.html).
5. D. Moore et al., “Inside the Slammer Worm,” *IEEE Security & Privacy*, vol. 1, no. 4, 2003, pp. 33-39.
6. S. Staniford, V. Paxson, and N. Weaver, “How to Own the Internet in Your Spare Time,” *Proc. 11th Usenix Security Symp.*, Usenix Assoc., 2002, pp. 149-167.
7. D. Moore et al., “Internet Quarantine: Requirements for Containing Self-Propagating Code,” *Proc. IEEE Infocom 2003, 22nd Ann. Joint Conf. IEEE Computer and Comm. Societies*, vol. 3, IEEE Press, 2003, pp. 1901-1910.
8. J.P. Anderson, “Computer Security Threat Monitoring and Surveillance,” tech. report, James P. Anderson Co., Fort Washington, Pa., 1980.
9. D.E. Denning, “An Intrusion Detection Model,” *IEEE Trans. Software Eng.*, vol. 13, no. 2, 1987, pp. 222-232.
10. D.M. Chess, “Tools & Techniques: Virus Verification and Removal,” *Virus Bull.*, Nov. 1991, pp. 7-11; [www.virusbtn.com/magazine/archives/pdf/1991/199111.PDF](http://www.virusbtn.com/magazine/archives/pdf/1991/199111.PDF).
11. J.O. Kephart, “A Biologically Inspired Immune System for Computers,” *Proc. 14th Int’l Joint Conf. Artificial Intelligence*, Morgan Kaufmann, 1995, pp. 20-25.
12. S. Forrest, S.A. Hofmeyr, and A. Somayaji, “Computer Immunology,” *Comm. ACM*, vol. 40, no. 10, 1997, pp. 88-96.

*Thomas M. Chen is an associate professor in the Department of Electrical Engineering at Southern Methodist University. His research interests include Internet security, particularly with respect to worm epidemics and intrusion detection; network traffic control; and real-time traffic-flow metering. Chen received a PhD in electrical engineering from the University of California, Berkeley. He is a senior member of the IEEE. Contact him at [tchen@engr.smu.edu](mailto:tchen@engr.smu.edu).*

*Jean-Marc Robert is a principal security researcher at Alcatel Canada Inc. in Ottawa, Ontario. His research interests are network and telecom infrastructure security, focusing mainly on denial-of-service attacks and worm propagation. Robert received a PhD in computer science from McGill University. He is a member of the IEEE. Contact him at [jean-marc.robert@alcatel.com](mailto:jean-marc.robert@alcatel.com).*

# Computer Wants You

**Computer is always looking for interesting editorial content. In addition to our theme articles, we have other feature sections such as Perspectives, Computing Practices, and Research Features as well as numerous columns to which you can contribute. Check out our author guidelines at**

**[www.computer.org/computer/author.htm](http://www.computer.org/computer/author.htm)**

**for more information about how to contribute to your magazine.**

Innovative Technology for Computer Professionals  
**Computer**