Editors: Elias Levy, aleph1@securityfocus.com
Iván Arce, ivan.arce@coresecurity.com

The Spread of the Witty Worm

Colleen Shannon and David Moore
Cooperative Association for Internet Data Analysis (CAIDA)

On Friday, 19 March 2004, at approximately 8:45 p.m. Pacific Standard Time (PST), an Internet worm began to spread, targeting a buffer overflow vulnerability in several Internet Security Systems (ISS) products, including its RealSecure Network, RealSecure Server Sensor, RealSecure Desktop, and BlackICE. The worm took advantage of a security flaw in these firewall applications that eEye Digital Security discovered earlier in March. Once the Witty worm—so called because its payload contained the phrase, "(^.^) insert witty message here (^,^)"—infects a computer, it deletes a randomly chosen section of the hard drive, which, over time, renders the machine unusable. (See the "Further information" sidebar on p. 50 for resources.)

While the Witty worm is only the latest in a string of self-propagating remote exploits, it distinguishes itself through several interesting features:

- It was the first widely propagated Internet worm to carry a destructive payload.
- It started in an organized manner with an order of magnitude more ground-zero hosts than any previous worm.
- It represents the shortest known interval between vulnerability disclosure and worm release—it began spreading the day after the ISS vulnerability was publicized.
- It spread through a host population in which every compromised host was proactive in securing its computers and networks.
- It spread through a population almost an order of magnitude smaller than that of previous worms, demonstrating worms' viability as an automated mechanism to rapidly compromise machines on the Internet, even in niches without a software monopoly.

In this article, we share a global view of the worm's spread, with particular attention to these worrisome features.

Witty worm background

In this section, we detail the technical aspects of the code and monitoring of the Witty worm.

Network telescope

Because Internet worm victims span diverse geographic and topological locations, the overall impact of a worm is difficult to measure from a single viewpoint. The University of California, San Diego (UCSD) Network Telescope (see the sidebar) consists of a large piece of globally announced IPv4 address space that we have instrumented to monitor network security events. The telescope contains almost no legitimate hosts, so inbound traffic to nonexistent machines is always anomalous in some way. Because the network telescope contains approximately 1/256th of all IPv4 addresses, we receive roughly one out of every 256 packets sent by an Internet worm with an unbiased random number generator.

Because we are uniquely situated to receive traffic from every worm-infected host, we provide a global view of the spread of many Internet worms.

ISS vulnerability

Several ISS firewall products contain a protocol analysis module (PAM) to monitor application traffic. The PAM routine in version 3.6.16 of iss-pam1.dll analyzed ICQ server traffic and assumes that incoming packets on port 4000 are ICQv5 server responses and that this code contains a series of buffer overflow vulnerabilities. eEye discovered this vulnerability on 8 March 2004 and announced it with ISS 10 days later. ISS released an alert, warning users of a possibly exploitable security hole and providing updated software versions that were not vulnerable to the buffer overflow attack.

Witty worm details

Once Witty infects a host, the host sends 20,000 packets by generating packets with a random destination IP address, a random size between 796 and 1,307 bytes, and a destination port. The packets' small size ensures fast transmission; as well, they are unlikely to be too big, for any network segment that they traverse—if they were too big, they would be broken into smaller pieces in the network, which could slow the spread of the worm. Because the Witty packets sizes are random, they are more difficult to filter than fixed

sized packets and complicate simple blocking measures that limit or prevent the worm's spread. If they had been a fixed size, it would have been easier to quickly block Witty traffic to limit or prevent the worm's spread. The worm payload of 637 bytes is padded with data from system memory to fill the random size, and a packet is sent out from source port 4000. After Witty sends the 20,000 packets, it seeks out a random point on the hard disk and writes 65kbytes of data from the beginning of iss-pam1.dll to the disk. After closing the disk, Witty repeats this process until the machine is rebooted or until it permanently crashes.

Witty worm spread

The perpetrators of previous Internet worms, including Code Red, Nimda, and SQL Slammer seeded a few hosts, which then proceeded to spread to the rest of the vulnerable population. The spread was slow early on—but then accelerated dramatically as the number of infected machines spewing worm packets to the rest of the Internet rose. Eventually, as the victim population became saturated, its spread slowed because there were few vulnerable machines left to compromise. Plotted on a graph, this worm growth appears as a sigmoid, an s-shaped exponential growth curve.

At 8:45:18 p.m. PST on 19 March 2004, the network telescope received its first Witty worm packet. In contrast with previous worms' spread dynamics, we observed 110 hosts infected in the first 10 seconds, and 160 at the end of 30 seconds. Because only about 12,000 computers were susceptible to witty out of a global IP address pool of about 4.3 million, the chances of a single instance of the worm infecting 110 machines so quickly are vanishingly small— worse than 10^{-607}. This rapid onset indicates that the worm used either a hitlist or previously compromised vulnerable hosts to start the worm's spread. In Figure 1, the initial vertical line shows preselected hosts coming online and then a transition to much slower growth thereafter.

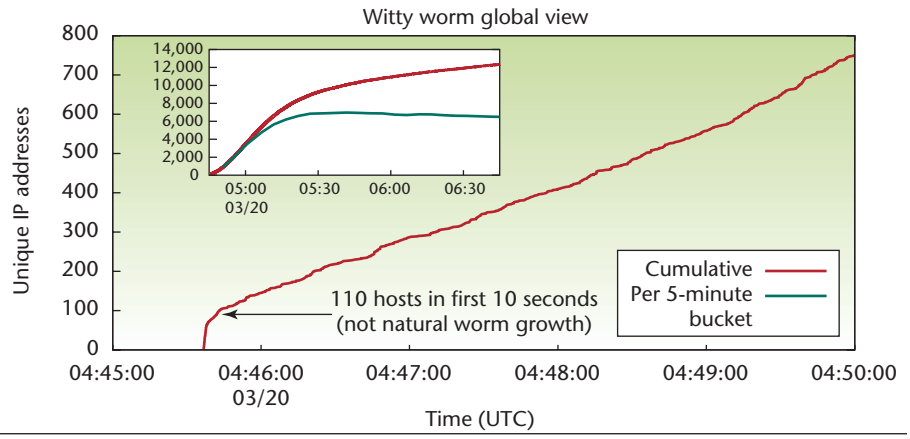

Figure 1. The Witty worm's spread. After the first minute, Witty followed an expected sigmoid curve. The inset shows the first two hours of Witty's spread. The number of active machines per five minutes (green line) stabilized after 45 minutes, indicating that almost all the vulnerable machines had been compromised. After that point, dynamic addressing (for example, Dynamic Host Configuration Protocol) caused the cumulative IP address total (the red line) to continue to rise. We estimate the total number of hosts infected by the Witty worm to be 12,000 at most.

After the sharp rise in initial coordinated activity, Witty followed a normal exponential growth curve for a pathogen spreading in a fixed population, reaching its peak after approximately 45 minutes, at which point the majority of vulnerable hosts had been infected. After this point, the churn caused by dynamic addressing—a single computer changing its IP address over time, typically via Dynamic Host Configuration Protocol (DHCP), or bridging between internal and external networks that can let multiple computers share one IP address, often using Network Address Translation (NAT)—causes the IP address count to inflate without any additional Witty infections. At the peak of the infection, Witty hosts flooded the Internet with more than 90 Gbits per second of traffic, sending more than 11 million packets per second.

Witty infected only about 1/10th as many hosts as the next smallest widespread Internet worm. While SQL Slammer infected between 75,000 and 100,000 computers, the vulnerable population of the Witty worm was only about 12,000 computers. Although researchers[1–3] have long predicted that a fast-probing worm could infect a small population very quickly, Witty is the first worm to demonstrate this capability. Witty took 30 minutes longer than SQL Slammer to infect its vulnerable population, but both worms spread far faster than human intervention could stop them. In the past, users of software that is not ubiquitously deployed have considered themselves relatively safe from most network-based pathogens. Witty demonstrates that a remotely accessible bug in any minimally popular piece of software can be successfully exploited by an automated attack.

Witty's destructive payload, in combination with efforts to filter its traffic and patch infected machines, led to rapid decay in the number of infected hosts (see Figure 2). Twelve hours after the worm began to spread, half of the Witty hosts were already inactive.

Witty worm victims

The vulnerable host population pool for the Witty worm was quite differ-

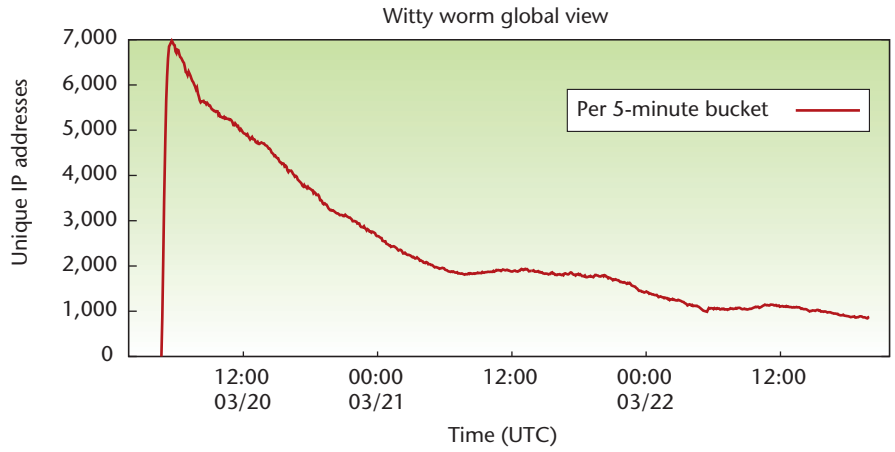

Figure 2. The number of unique hosts infected with the Witty worm over time. Infected Witty hosts were deactivated much more quickly than with previous worms. Although prompt network filtering and active cleanup of compromised hosts played an important role, we believe that the rapid decay in the number of hosts actively spreading Witty was primarily due to the destructive payload crashing infected machines.

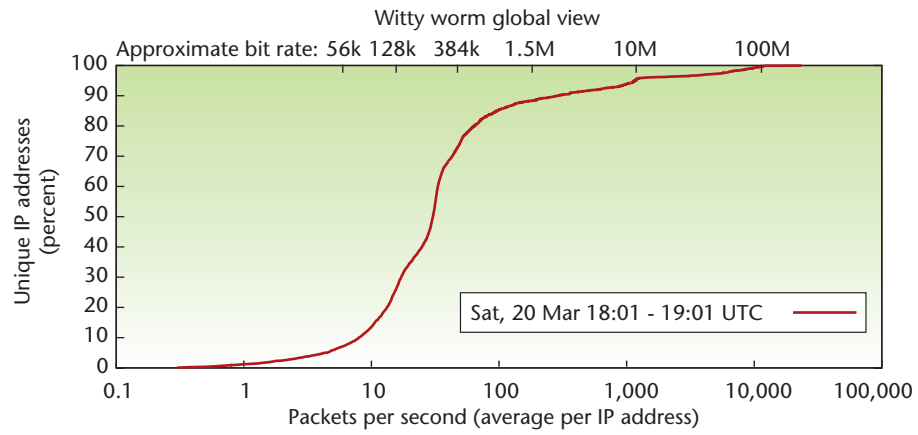

Figure 3. The scanning rate in packets-per-second of hosts the Witty worm infected. The connection bandwidths that correspond to the packet rates are marked along the top of the graph. Fifty-three percent of infected hosts had connection speeds between 128 kbps (15 pps) and 512 kbps (60 pps). The maximum packet rate observed from one host was 23,500 pps sustained for at least one hour.

ent from that of previous virulent worms. Previous worms have lagged several weeks behind publication of details about the remote-exploit bug, and large portions of the victim populations appeared not to know what software was running on their machines, let alone take steps to make sure that software had up-to-date security patches. In contrast, Witty infected a host population that was proactive about security—they were running firewall software. The Witty worm also started to spread the day after information about the exploit—and the software upgrades to fix it—were available.

Like SQL Slammer, the Witty worm was bandwidth limited—each infected host sent packets as fast as its Internet connection could transmit them. As Figure 3 shows, Witty infected a relatively well-connected pool of hosts. Sixty-one percent of infected hosts transmitted at speeds between 96 kbps (11.2 pps) and 512 kbps (60 pps). The average speed of an infected host was 3 Mbps (357 pps), although during the peak of the worm's spread, the average speed reached 8 Mbps (970 pps). We also observed 38 machines transmitting Witty packets at rates over 80 Mbps continuously for more than an hour.

Some of the most rapidly transmitting IP addresses might actually be a larger collection of hosts behind a Network Address Translation (NAT) device of some kind. By infecting firewall devices, Witty proved particularly adept at thwarting security measures and successfully infecting hosts on internal networks. We also observed more than 300 hosts in the first few hours transmitting Witty from source ports other than 4000. Because the defining characteristic for successful Witty infection is a source port 4000 packet, presumably these machines are NAT boxes rewriting the source port of packets originating at downstream infected hosts. Sixty-seven of those NAT boxes also sent Witty packets with the correct source port 4000, so while some NATs might have artificially inflated the a single infected host's transmission speeds, others might have artificially deflated them by spreading traffic across other ports.

Witty worm hosts showed a wide range of infection durations (see Figure 4). Several factors influence our measurements of infection duration:

- Dynamic addressing significantly affects the amount of time an IP address remains active. As with the SQL Slammer worm, the flood of packets from an infected host can reset its upstream connection (particularly with dial-up hosts), causing the host to disconnect and reconnect from a different IP address.
- End users might be unaware that a worm is causing their computer or Internet connection to be slow, and they might reboot their computers

in the hope that this will fix the problem. If the random disk writes have not damaged anything critical to the boot process, each host could receive a different dynamic address after the reboot. For these dynamically addressed hosts, the observation of the infection duration reflects the duration for which each host maintained its Dynamic Host Configuration Protocol (DHCP) lease, rather than the true duration of infection on that host.
- Traffic filtering also artificially limits our view of a host's infection duration, but at least in this case, we accurately record the duration for which the victim spread the worm to other vulnerable hosts.
- Witty carried a destructive payload that would eventually crash the infected machine. Thus even without a dynamic address or any human intervention, Witty would eventually (and often permanently) deactivate each infected host.

Because US-based ISS is a much smaller company than Microsoft, with less extensive overseas operations, the majority of Witty worm infections occurred in the US. Table 1 shows the breakdown of infected machines by country. The hostnames of 33 percent of all infected machines inhabited the .com domain, while the .net domain contained 20 percent of all infected computers, and the .edu domain sourced 1 percent of all Witty infections.

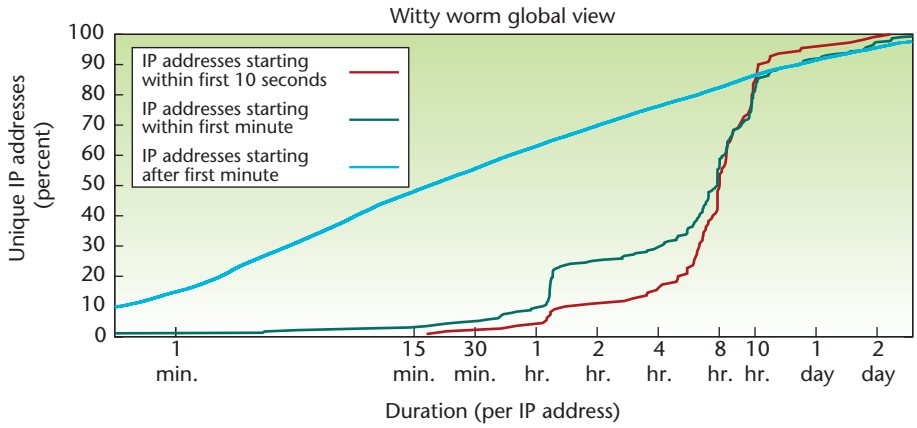

Figure 4. The infection duration for Witty's hosts. Unlike previous widespread Internet worms, the worm payload's malicious disk writes (which crashed infected computers) curtailed Witty hosts' infection duration. In-network filtering and active host cleanup also played important roles in limiting the Witty worm's spread.

The Witty worm incorporates several dangerous characteristics. It was the first widely spreading Internet worm to actively damage infected machines. As noted earlier, it started from a large set of machines simultaneously, indicating the use of a hitlist or a large number of compromised machines. Witty demonstrated that any minimally deployed piece of software with a remotely exploitable bug can be a vector for wide-scale compromise of host machines without any action on the victim's part. The practical implications of this are staggering: with minimal skill, a malevolent individual could break into thousands of machines and use them for almost any purpose with little evidence of the perpetrator left on most of the compromised hosts.

While many of these Witty features are novel in a high-profile worm, the same virulence combined with greater potential for host damage has been a feature of bot networks (botnets) for years. Any vulnerability or backdoor that a worm can exploit can also be exploited by a vastly stealthier botnet. While all of the worms seen thus far have carried a single payload, bot functionality can be easily changed over time. Thus, while worms are a serious threat to Internet users, botnets' capabilities and stealth make them a more sinister menace. The line separating worms from bot software is already blurry; over time we can expect to see increasing stealth and flexibility in Internet worms.

Witty was the first widespread Internet worm to attack a security product. While technically the use of a buffer overflow exploit is commonplace, the fact that all victims were compromised via their firewall software the day after a vulnerability in that software was publicized indicates that the security model in which end users apply patches to plug security holes is not viable.

It is both impractical and unwise to expect every individual with a computer connected to the Internet to be a security expert. Yet the current mechanism for dealing with security holes expects end users to constantly monitor security alert Web sites to learn about security flaws and then immediately download and install patches. Patch installation is often difficult, involving a series of complex steps that must be applied in a precise order.

The patch model for Internet security has failed spectacularly. To remedy this, there have been many

Table 1. Geographic distribution of Witty worm victims by country.

COUNTRY	PERCENT
United States	26.28
United Kingdom	7.27
Canada	3.46
China	3.36
France	2.94
Japan	2.17
Australia	1.83
Germany	1.82
Netherlands	1.36
Korea	1.21

Further information

For more information about the UCSD Network Telescope and other Internet worms, see www.caida.org/analysis/security/witty/#Info. Animations of the spread of the worm in the US and throughout the world are also available at www.caida.org/analysis/security/witty/#Animations.

Additional resources include:

- The original publication of the ISS buffer overflow vulnerability (a joint release with ISS)—eEye's *Internet Security Systems Pam ICQ Server Response Processing Vulnerability*, www.eeye.com/html/Research/Advisories/AD20040318.html.
- The ISS version of the initial release of the vulnerability in their products (a joint release with eEye)—Internet Security Systems, *Vulnerability in ICQ Parsing in ISS Products*, http://xforce.iss.net/xforce/alerts/id/166.
- Symantec's threat analysis and mitigation information for the Witty Worm—Symantec, W32.Witty.Worm, http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html.
- An analysis of the Witty worm functionality—LURHQ Threat Intelligence Group, Witty Worm Analysis, www.lurhq.com/witty.html.
- A dissassembly of Witty worm code via BugTraq—K. Kortchinsky, *Black Ice Worm Disassembly*; www.caida.org/analysis/security/witty/BlackIceWorm.html.
- More information about the design and implementation of the UCSD network telescope—David Moore et al., *Network Telescopes: Observing Small or Distant Security Events*, www.caida.org/outreach/presentations/2002/usenix_sec/.
- Dan Geer's Workshop on Economics and Information Security presentation including data motivating the case for end-user liability—www.dtc.umn.edu/weis2004/weis-geer.pdf.
- Educated Guesswork blog summary of Geer's remarks about end-user liability—www.rtfm.com/movabletype/archives/2004_05.html#000907.

suggestions for ways to try to shoehorn end users into becoming security experts, including making them financially liable for the consequences of their computers being hijacked by malware or miscreants. (see the "Further information" sidebar)

Notwithstanding the fundamental inequities involved in encouraging people to sign on to the Internet with a single click, and then requiring them to fix flaws in software marketed to them as secure with technical skills they do not possess, many users do choose to protect themselves at their own expense by purchasing antivirus and firewall software. Making this choice is the gold-standard for end-user behavior—they recognize both that security is important and that they do not possess the skills necessary to effect it themselves. When end users participating in the best security practice that can be reasonably expected get infected with a virulent and damaging worm, we must reconsider the notion that end-user behavior can solve or even effectively mitigate the malicious software problem and turn our attention toward both preventing software vulnerabilities in the first place and developing large-scale, robust, and reliable infrastructure that can mitigate current security problems without relying on end-user intervention. □

Acknowledgments

The network telescope and associated security efforts are a joint project of the University of California, San Diego (UCSD) Computer Science and Engineering Department and the Cooperative Association for Internet Data Analysis (CAIDA). We thank Brian Kantor, Jim Madden, and Pat Wilson of UCSD for technical support of the Network Telescope project; Mike Gannis, Nicholas Weaver, Wendy Garvin, Team Cymru, and Stefan Savage for feedback on this document; and the Cisco Product Security Incident Response Team(PSIRT), Wendy Garvin, Team Cymru, Nicholas Weaver, and Vern Paxson for discussion as events unfolded. The Cisco Systems University Research Program, the US National Science Foundation, DARPA, the US Department of Homeland Security, and CAIDA members provided support for this work.

References

1. D. Moore et al., "Inside the Slammer Worm," *IEEE Security & Privacy*, vol. 1, no. 4, 2003, pp. 33–39; www.computer.org/security/v1n4/j4wea.htm.
2. S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," *Proc. 11th Usenix Security Symp.* (Security'02), Usenix Assoc., 2002; www.icir.org/vern/papers/cdc-usenix-sec02/.
3. D. Moore et al., "Internet Quarantine: Requirements for Containing Self-Propagating Code," *Proc. 22nd IEEE Infocom*, 2003, IEEE CS Press, 2003; www.cs.ucsd.edu/users/savage/papers/Infocom03.pdf.

Colleen Shannon *is a senior security researcher at the Cooperative Association for Internet Data Analysis (CAIDA) at the San Diego Supercomputer Center (SDSC) at the University of California, San Diego (UCSD). She is currently working toward her BS in biology from UCSD. Her research interests focus on network security, including wide-area traffic measurement, data collection and distribution, network application identification, and risk analysis. Contact her at cshannon@caida.org*

David Moore *is the assistant director of CAIDA and a PhD candidate in the UCSD Computer Science Department. He received a BS in computer science and a BA in mathematics from UCSD. His research interests include high-speed network monitoring, denial-of-service attacks and infrastructure security, and Internet traffic characterization. Contact him at dmoore@caida.org.*