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

Worm Propagation and Generic Attacks

This past December, a new family of worms was discovered. The family, Santy, attacked Web applications written in the PHP scripting language. Santy is interesting for two reasons: First, its worms used Web search engines to locate likely targets; second, a Santy variant exploited a generic flaw in PHP applications, rather than a specific vulnerability.

The defining task of propagating malicious code is to locate new targets to attack. Viruses search for files in a computer system to which to attach, whereas worms search for new targets to which to transmit themselves. Depending on their method of transmission, malicious code writers have developed different strategies for finding new victims.

Finding new targets

Worms transmitted via email have had great success propagating themselves because they find their next targets either by raiding a user's email address book or by searching through the user's mailbox. Such addresses are almost certain to be valid, permitting the worm to hijack the user's social web and exploit trust relationships. In most cases, the worm will craft its own message to send to the target, but some will wait for the user to send a message and attach themselves to it.

Network worms, those that attack network services, must determine their next victim's IP address. When such worms first emerged, almost all of them simply generated target IP addresses at random. This technique wasn't efficient for two reasons. First, the IP version 4 address space is reasonably large (2^{32}). Second, the IP address space is sparsely populated, with hosts clustered in some areas, while the rest remains unpopulated. Through the years, authors of network worms have devised several strategies to more effectively locate new targets. For instance, some worms, such as Code Red II, displayed a bias toward local addresses—those near the current computer in the IP address space—which usually implies closeness in the network topology. The reasoning behind this is that there's likely to be other machines near the infected machine, and furthermore, they're likely to be running similar software.

These developments led to several research papers that examine the efficiency of network worm propagation, such as Tom Vogt's *Simulating and Optimising Worm Propagation Algorithms* (http://web.lemuria.org/security/WormPropagation.pdf) and papers that speculate on future target selection algorithms, such as Stuart Staniford, Vern Paxson, and Nicholas Weaver's *How to Own the Internet in Your Spare Time*.[1] The latter coined the terms *Warhol worm* and *flash worm* to describe worms with

Elias Levy
Symantec

Continuing trends

In the September/October 2003 edition of Attack Trends ("The Rise of the Gadgets," pp. 78–81), my coeditor Iván Arce examined threats posed by the emergence of networkable devices. A year later, in the September/October 2004 Attack Trends installment ("The Shellcode Generation," pp. 72–76), we reported the first known cell phone worm, Cabir, which attacked devices that used the Symbian operating system and propagated via the Bluetooth networking technology. The trend continues with the discovery of more malicious code that can infect mobile phones.

Since Cabir's discovery, its source code has become available on the Internet, and several variants have appeared. In addition, a few Trojans, such as Skulls and Locknut, have come to light, which disable all functionality from the phone other than making and receiving calls.

In the July/August 2004 edition of Attack Trends ("Approaching Zero," pp. 65–66), I discussed the increase of zero-day vulnerabilities—vulnerabilities found exploited in the wild without security researchers' or vendors' previous knowledge of them—as the result of illegal economic incentives. In August 2004, soon after the article was published, an attack against a zero-day vulnerability in the Winamp media player was found in the wild. It was reported in the Bugtraq mailing list that the attack, which appeared on several Web sites, installed spyware, Trojans, and other malicious code in a victim's computer.

certain types of target-selection algorithms that, theoretically, could infect the Internet within minutes.

Malicious code that propagates by attacking Web applications requires different targeting algorithms because uniform resource identifiers (URIs) represent a unique address space; the URI's path portion is often static, its query portion is usually known for a specific vulnerability (although not always), and its protocol portion is usually HTTP.

One approach to the problem of locating vulnerable Web pages is to ignore the URI's host portion, attempt to connect to random hosts on port 80, and, if successful, attempt an attack on the host, hoping that the server has a vulnerable default site that doesn't require an HTTP host header. Alternatively, the malicious code could act as a Web crawler and attack new sites as they're discovered by parsing Web pages in its search for links. But current Web crawlers make their data available to anyone, so why replicate their work? We call these Web crawlers search engines.

Web-searching worms

A search engine is a quick and easy way for malicious code to locate new targets. Yet, using it also has its downsides. The order of the results returned is usually static for a given query in a short time period; to get around this the malware must come up with a way to randomize the order of results, so that multiple instances of the same malware don't try to attack the same targets in the same order, thereby wasting resources. In addition, the search engine can become a single point of failure. Search engine operators can choose not to return results for the malicious code's query or even purge the search engine database of Web pages that match the query if they discover its being used for malicious purposes.

The Santy family of malicious code is the first known to attack Web applications and that uses a Web search engine to locate new targets. The Security Response Web site has more on these worms at http://securityresponse.symantec.com/avcenter/. In particular, the Perl.Santy.A (.../venc/data/perl.santy.html) and Perl.Santy.C (.../venc/data/perl.santy.c.html) worms used the Google search engine to locate new targets, Perl.Santy.B (.../venc/data/perl.santy.b.html) used the AOL and Yahoo search engines, and Perl.Lexac (.../venc/data/perl.lexac.html), a Santy variant, used both Google and Yahoo search engines. Other than Perl.Lexac, the Santy worms were searching for a Web page in phpBB, a bulletin board system written in PHP, that was vulnerable to the PHPBB Viewtopic.PHP PHP Script Injection Vulnerability (BID 10701; www.securityfocus.com/bid/10701).

Although Santy is unique, at least one other worm family, Mydoom, has used Web search engines to find new targets, although they didn't attack Web applications. Instead, they were mass mailer worms that searched the Web for email addresses. The first known variant of Mydoom to include this feature was W32.Mydoom.M@mm (http://securityresponse.symantec.com/avcenter/venc/data/w32.mydoom.m@mm.html), which queried AltaVista, Google, Lycos, and Yahoo. These worm queries placed a significant burden on these search engines.

Vulnerability-finding worms

The second interesting aspect of the Santy family is displayed by the Perl.Lexac worm: it didn't exploit a specific known vulnerability in a particular Web application, but instead exploited a generic vulnerability in Web applications written in PHP.

The PHP language has an operation that allows an application to "include" a file—that is, the file's contents will parse as PHP and execute in the context of the application. An odd feature of this operation is that by passing a URL to the operation, you can ask it to include a file located at a remote location. A common security error in PHP applications is to include a file whose name is passed to the application as a parameter in the HTTP request, without first validating that the file is trustworthy. Thus, remote adversaries can tell PHP applications to include malicious PHP code that's available in a Web site. (For more on this, see PHP Security Mistakes; www.devshed.com/c/a/PHP/PHP-Security-Mistakes.)

This generic vulnerability is what Perl.Lexac exploited. It searched the Web for any PHP Web pages by working through Google, and then proceeded to parse Web page URLs; if any contained any query parameters, it attempted to access them multiple times, each time replacing one of the query parameter values with a URL that returned malicious PHP code that would infect the computer. Most such requests failed, but a sufficient number of PHP applications suffer from this vulnerability, thereby successfully infecting a substantial number of machines. At a basic level, Perl.Lexac acts like Web application security assessment tools. It determines the applications' inputs and then performs testing by modifying them.

Amichai Schulman, chief technical officer of Imperva, predicted these type of worms that target generic vulnerabilities in his white paper *Web Application Worms: Myth or Reality?* (www.imperva.com/download.asp?id=1). As he notes in the paper, a type of vulnerability that readily lends itself to this type of generic vulnerability attack is SQL injection vulnerabilities in Web applications—vulnerabilities that arise from the improper creation of SQL queries based on untrusted input, which leads to adversaries executing SQL statements in the context of the Web application. As the low level causes of SQL injection vulnerabilities are well known and few in number, and the possible avenues for executing arbitrary code within SQL largely independent of table structure, these vulnerabilities lend themselves to easy discovery and exploitation automation.

Worm authors will continue to research new methods to more efficiently let their creations propagate, and if we are to believe the research in this area, worms that can infect the entire Internet in minutes don't allow for sufficient time to act after the fact. Therefore, we must concentrate our efforts toward finding and deploying proactive security defenses.

The idea of worms that attack generic vulnerabilities is a frightening one. The ultimate solution to worm outbreaks is to patch vulnerable systems. When the vulnerability is specific, we have a known set of applications that we must fix, but when the vulnerability is generic, any number of applications could be vulnerable—finding them and fixing them can be a difficult task. Therefore, we must also concentrate on finding solutions to whole classes of vulnerabilities, rather than focusing on fixing individual problems within the usual patch cycle. ▢

Reference

1. 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/.

Elias Levy *is an architect with Symantec. Previously, he was the chief technology officer and cofounder of SecurityFocus and the moderator of Bugtraq, a vulnerability disclosure mailing list. His research interests include buffer overflows and networking protocol vulnerabilities. He is also a frequent commentator on computer security issues and participates as a technical advisor to a number of security-related companies. Contact him at aleph1@securityfocus.com.*