Dowgirl.com...

Buffer Overflow Debuts on CNN

Summary: The speed at which buffer overflow exploits can propagate, the enormous explosion of the use of new devices (music sharing, pod casting, video downloading) and the blurring of the line between what is and isn't a computer make buffer overflow a continuing problem for IT security. Effective controls for this type of attack require sophisticated systems, which can recognize the attack in progress and drop machines and networks before they can compromise the system. It is clear that the zero-day exploit is a reality, which we will all face sooner rather than later, and I am betting that it will be a buffer overflow exploit.

Buffer Overflow Debuts on CNN
October 2005, Dowgirl

Dateline: August 16, 2005 – Starting at about 5:30 PM, CNN devoted its entire news coverage to a computer worm infecting numerous computers at the 24-hour news network. Whether CNN was unable to broadcast its planned news line-up due to the computer issues, or whether they simply choose to devote 90 minutes of prime air time to the story isn't clear. What is clear was the image of Wolf Blitzer showing the world how his laptop repeatedly loaded the Windows desktop, displayed an alert message, shutdown and rebooted.

The worm that invaded systems at CNN, The New York Times and others exploited a vulnerability classified as a buffer overflow. While buffer overflow sounds like a brute force, data intensive system attack – we are all sensitive to issues surrounding our computers' memory – buffer overflows are in fact carefully crafted attacks. They exploit vulnerabilities at the core of how modern computers and computer programmers work.

A buffer is a contiguous block of computer memory that temporarily holds data, and programs are constantly creating, using and then releasing blocks of memory as they operate and execute commands. All computers, regardless of the operating system, work by "pushing" and "popping" data in and out of this area of memory called the stack. One key piece of data that is pushed and popped into the buffer involves the address, or physical location, of data and programming code.

As buffers are dynamically created and then overwritten, it is necessary to pass instructions to the computer about how much memory to allot for any particular buffer. If more information is passed to the buffer than it was designed to handle, it overflows and data in the buffer is overwritten. In nearly all instances, this overwriting causes an error and the application fails. However, it is possible to overflow the part of the buffer that stores the address and forcing execution of code that resides at the new, bogus address – usually somewhere else in the overwritten buffer.

UNIX operating systems, with open source codes, were the first systems attacked through buffer overload, because hackers require detailed knowledge of how individual commands use and create buffers. The coding was initially more complex for attacking Windows platforms, and progress was hampered by the lack of access to the Windows source codes. But Windows vulnerabilities are numerous and many hackers have been aggressively targeting Windows for years. A list by a major computer security company of the current, most critical vulnerabilities indicates that 4 out of the top 5 are buffer overflows, including attacks on Window's plug and play service, print spooler, ASN library and RPC message service. Every day it seems like Windows has brought users and hackers a new way to interact (and overflow buffers.)

Programmers, too, are to blame for the increasing rise in buffer overflow exploits. Writing code is a tedious and time-consuming task, where even one character out of place can prevent a program from running, so programmers frequently modify or adapt existing code to create a new program; programmers rarely start from scratch. Entirely new programs and applications often have at their core older, legacy coding. In most instances newer, robust commands are available that check for and reject data that would overflow the buffer, but the use of legacy coding means that many applications are still using the older, exploitable commands. Lurking at the heart of a staggering number of applications today are the seeds of buffer overflow – in the PC, in the server, firewall, router and other networking devices. Getting to market first is a major business driver for computer device companies, and going back to check and code the devices from scratch just isn't going to happen. Remember, buffer overflow is a possibility with nearly any type of device using dynamic memory, due to the physical structure of the CPU.

Buffer overflow exploits are increasingly becoming more sophisticated and worrisome to the IT security community. January 25th, 2003, the fastest spreading worm to date began infecting machines running Microsoft SQL Server or MSDE 2000 (Microsoft SQL Server Desktop Engine). Within 10 minutes, the worm had infected 90% of vulnerable hosts, over 75,000 computers. At its peak, the worm was doubling the total number of infected machines every 8.5 seconds.

March 19, 2004, an Internet worm (Witty) began to spread, targeting a buffer overflow vulnerability in several Internet Security Systems (ISS) products. The worm takes advantage of a security flaw in firewall applications that was discovered earlier that March. Once the Witty worm infects a computer, it deletes a randomly chosen section of the hard drive, over time rendering the machine unusable.

Witty is one of the first widely propagated Internet worms to carry a destructive payload and represented (at the time) the shortest known interval between vulnerability disclosure and worm release -- it began to spread the day after the ISS vulnerability was publicized. Witty also infected 110 hosts in its first 10 seconds in what appeared to be a well-planned, targeted attack.

What gets the most attention from IT professionals is that Witty targeted the firewall, and that the worm successfully attacked systems with superior security. Witty did come after a patch was issued, but just 36 hours after, and the worm was well-written, professional grade code. We are fast approaching the true zero-day exploit. (A zero-day exploit is one that takes advantage of security vulnerability on the same day that the vulnerability becomes generally known.)

Now security vulnerability researchers are predicting the eventual emergence of router worms: malware designed to spread from router to router, bringing down vast segments of network infrastructure. Although the idea has existed as a possibility for almost two decades, no actual instances have yet occurred, but 5 years ago it was rare to see a buffer overload exploit in a Windows platform, and look where we are now.

To get an idea of how concerned companies are of router worms, consider the brouhaha that was generated earlier this summer at the Black Hat security conference in Las Vegas, when Cisco threatened legal action to stop the conference's organizers from allowing a 24-year-old researcher from a rival tech firm to discuss how he says hackers could seize control of Cisco's Internet routers, which dominate the market. The paper, a presentation called "The Holy Grail: Cisco IOS Shellcode Remote Execution" was presented, and instead of squelching the information, copies of the presentation are widely available on the internet from IT organizations which strongly believe that disclosing security vulnerabilities are our best mechanisms for improving security.

The speed at which buffer overflow exploits can propagate, the enormous explosion of the use of new devices (music sharing, pod casting, video downloading) and the blurring of the line between what is and isn't a computer make buffer overflow a continuing problem for IT security. Effective controls for this type of attack require sophisticated systems, which can recognize the attack in progress and drop machines and networks before they can compromise the system. It is clear that the zero-day exploit is a reality, which we will all face sooner rather than later, and I am betting that it will be a buffer overflow exploit.

Page Lef t Intentionally Blank