

Buffer Overflows - What Are They and What Can I Do About Them?

Buffer overflows have been a problem in software-based systems and applications for a long time. One of the first significant computer break-ins that took advantage of a buffer overflow was the Morris worm, and that happened in November 1988. The worm took advantage of a buffer overflow in the finger service, a service that dispenses information about the set of users logged into a UNIX-based computer system.

Even though the cause was highly publicized, buffer overflows are still a major cause of intrusions today. In fact, the first six CERT® Coordination Center advisories issued in 2002 describe buffer overflow flaws in several common computer programs.

What are buffer overflows and why are they still a significant source of program vulnerabilities when the problem is well understood and the solutions well known? And what can Joe and Jane User do to prevent intruders from exploiting their computer systems? This article discusses those topics.

I've heard about buffer overflows, but I don't know what they are. Please explain.

Let's begin with an analogy. A buffer overflow is like trying to put ten pounds of sugar into a container that only holds five pounds. Once the container fills, the rest spills all over the counter and floor, making a mess that somebody has to clean up.

A buffer overflow is possible because the creator of a computer program wrote lines of code that do not properly check the size of the destination area or buffer—the five-pound container—to see if it is big enough to completely hold its new contents—the ten pounds of sugar. If the data intended for its new home doesn't fit and spills over, it too can create a mess that requires somebody's attention, usually the computer's owner or a system administrator.



However, a buffer overflow is only a problem if the buffer in question is overflowed. Until then, it is harmless.

When the sugar spills, the counter is covered. It can be wiped or vacuumed, returning the counter to its original condition. In contrast, when a buffer overflows, the excess information overwrites the previous contents of parts of a computer's memory space. Unless those overwritten contents were saved or can be recreated, they are lost forever.

Among the information lost is the ordered list of subroutines that had been called by the program up until the time when the buffer overflow occurred. In addition, the information given to

those subroutines—called arguments—is also lost. This means that the program can't find a way back through the subroutines it needs to complete its programmed task. It is like a man walking through a desert. If he relies on his trail of footprints in order to retrace his steps, he will be lost when a sandstorm erases them.

OK, so “sugar” spills all over the “counter,” and “footprints” are erased causing the man in the desert to lose his way. So what?

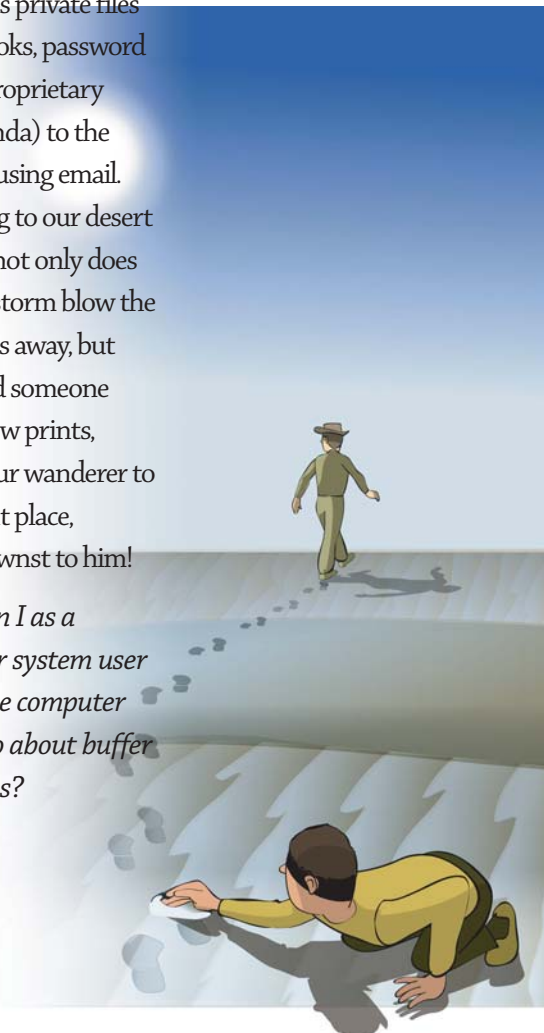
The problem is really worse than the program simply losing its way. The intruder overflows the buffer with a carefully crafted exploit script (a program written for malicious use) and then tells the program to treat the buffer as instructions and carry them out—execute them. The program does what the intruder wants, not what the programmer intended. In effect, the intruder becomes the programmer because his or her instructions are executed.

Intruders often reprogram an application to run a different program. For example, an intruder can start a new program that sends private files

(checkbooks, password files, or proprietary memoranda) to the intruder using email.

Returning to our desert analogy, not only does the sandstorm blow the footprints away, but afterward someone makes new prints, leading our wanderer to a different place, unbeknownst to him!

What can I as a computer system user and home computer owner do about buffer overflows?



Since buffer overflows are a programming problem, they can only be permanently fixed by repairing the broken program code. You probably don't have the source code for the applications you run, nor do you want that code along with the responsibility to fix each and every buffer overflow.

The best course of action is to monitor the web sites of your operating system and application vendors, applying any patches they provide as soon as you can. This can be tricky because installing a patch may cause another application to break, and you aren't likely to know this until you've installed the patch. Ask the vendor providing the patch if they know of any interoperability problems and also see if there is a way to remove a patch once it is applied. This gives you a recovery strategy should you need one.

Until there is a patch, you've got some decisions to make. If the affected software is optional, even temporarily, seriously consider removing it from your system. To use another analogy, the fewer doors and windows in your house, the fewer ways there are for an intruder to enter.

If it is not an option to remove the software and others who share your computer also use that software, you should limit who can use it by controlling access. Use the features of your operating system to define the set of users who can access the software. Create a group consisting of the users you trust, and assign the appropriate permissions to the group. Enable any logging and auditing your system provides so that you can track software use and monitor possible abuse.

If the software provides a service to the network, web service for example, you may not be able to limit access. You may have to live with the problem until your vendor can repair it. You need to compare the risk of your computer being broken into with the potential loss of business and revenue caused by temporarily halting the service. Once you've made that comparison, you need to decide which course of action to take.

I think I get it now. Buffer overflows make a mess and they can cause my computer's programs to lose their mind. I can patch them, remove them from my system, or accept the risk that an intruder won't find my computer until a patch becomes available.

Yes, that's a good summary. Good luck