acmqueue

Debugging on Live Systems

It's more of a social than a technical problem

Dear KV,

I've been trying to debug a problem on a system at work, but the control freaks who run our production systems don't want to give me access to the systems on which the bug always occurs. I haven't been able to reproduce the problem in the test environment on my desktop, but every day the bug happens on several production systems. I'm at the point of thinking about getting a key logger so I can steal the passwords necessary to get onto the production systems and finally see the problem "in the wild." I've never worked for such a bunch of fascists in my entire career.

Locked Down and Out

Dear Locked,

First of all, while most companies are inherently nondemocratic, few of them are fascist. Fascism went out of style sometime around 1945 and really hasn't made a comeback since. Secondly, I do sympathize—no one should be prevented from fixing a bug simply because of lack of access to the appropriate systems.

What many programmers and technical people fail to comprehend is that, as a colleague recently put it, "access implies responsibility." This is why the sudo program has the warning, stolen from the Spider-Man comics, "With great power comes great responsibility."

Debugging a program or a system can, and often does, have negative side effects, either by slowing down the system or changing the results of some calculation in an unintended fashion. The people who run your production systems are right to be wary of letting any random programmer loose in their domain. If you break something, it is likely to come down on their heads, and they'll have to fix it while you stand there glumly repeating, "Well, it wasn't supposed to do *that*!"

Your best bet is to try setting up a production system outside of the production environment first, as a test machine. I'm surprised by how many companies work without such staging machines, going directly from the developers' desktops to their production environments. If the bug won't happen without real workloads, then it's time to get a machine in the production environment sufficiently isolated so that it can be given a workload without destroying the machines that are doing productive work.

By now you might have noticed that this advice is less technical and more about social engineering. Programmers need to be willing to work with the people who have to keep systems up 24 hours a day, 7 days a week, if they want to be trusted enough to be able to debug live or near-live systems.

Two final thoughts: using a keyboard logger is not a way to gain trust, and telling someone in a public column that you're thinking about it is as dumb as tweeting your murder plans.

KV

Dear KV,

A program I've just been handed at work keeps crashing, and each time I look at it in the debugger and examine various bits of memory I see the pattern `0xdead0de` in different parts of allocated memory. Is this a joke? Do you think that my co-workers are hazing me?

OxDead Tired of this Code

Dear OxDead,

It is common practice for programmers to set memory to an easily recognizable value when they are trying to debug memory-smash bugs. You might think that they would clear all the bytes in the buffer to be `0x00`, but that doesn't help if some piece of code is writing NULL bytes all over your buffers. Using a known pattern such as `0xdead0de` makes it easier to find these problems in a debugger. As you've seen, you print a buffer and you see the pattern. If instead you saw, say, `0xde00c0de`, then you would know that someone had written a NULL byte in the middle of your memory. Maybe you wanted that, maybe you didn't, but now, at least, you can clearly see it. For extra cleverness points you can set a watchpoint—if it's supported by your hardware—which stops the program if some variable or part of memory does *not* equal `0xdead0de`. I tend to set buffers I'm debugging to be all `0x69`, because if I see that number, then I know it's my own personal bit of work.

For programmers who deal with network packets, a known pattern has another advantage. Most people write code on systems that are based on the Intel x86 architecture, which is known in network parlance as a little-endian system. A little-endian system stores the most significant byte of a multibyte word last. Network protocols are big endian, which is the opposite of how x86 processors store data in memory. All network programmers know the C macros `htonl()`, `ntohl()`, `htons()`, and `ntohs()`, which do the proper swapping of host-to-network endianness and back. A good way to debug a network protocol is to transmit data such as `0xdead0de` in the packets and then make sure it doesn't look like `0xdec0adde` when it arrives in your program's memory. Using this trick makes it easier to figure out where you might have left out a byte-swapping macro.

So, much as I would like to think that your co-workers are hazing you, it's far more likely that they are trying to be helpful.

KV

KODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who currently lives in New York City.

© 2011 ACM 1542-7730/11/0900 \$10.00