acmqueue

A Lesson in Resource Management

Waste not memory, want not memory—unless it doesn't matter

Dear KV,

I've been reworking a device driver for a high-end, high-performance networking card and I have a resource allocation problem. The devices I'm working with have several network ports, but these are not always in use; in fact, many of our customers use only one of the four available ports. It would greatly simplify the logic in my driver if I could allocate the resources for all the ports—no matter how many there are—when the device driver is first loaded into the system, instead of dealing with allocation whenever an administrator brings up an interface. I should point out that this device has a good deal of complexity and the resource allocation isn't as simple as a quick malloc of memory and pointer jiggling—a lot of moving parts are inside this thing.

We're not talking about a huge amount of memory by modern standards—perhaps a megabyte per port—but it still bothers me to waste memory, or really any resource, if it's not going to get used. I'm old enough to remember eight-bit computers with 64 kilobytes of RAM, and programming those gave me a strong internal incentive never to waste a byte, let alone a megabyte. When is it OK to allocate memory that might never be used, even if this might reduce the complexity of my code?

Fearful of Footprints

Dear Footprints,

The answer to your question is easy. It's sometimes OK to allocate memory that might never be used, and it's sometimes not OK to allocate the same memory. Ah, are those the screams of a programmer without a black-and-white, true-or-false answer to the question that I hear? Delightful!

Software engineering, much to your and my chagrin, is the study of tradeoffs. Time vs. complexity, expediency vs. quality—these are the choices we deal with every day. It's important for engineers to revisit their assumptions periodically, perhaps every year or two, as the systems we work on change under us quite quickly.

Programmers who are paying attention to the systems they use—and I know that each and every one of my readers is paying attention—have seen these systems change dramatically over the past five years, just as they had the five years before that, and so on, back to the first computers. While processor frequency scaling may have paused for the moment (and we'll see how long that moment lasts), the size of memory has continued to grow. It is not uncommon to see single servers with 64 and 128 gigabytes of RAM, and this explosion of available memory has led to some very poor programming practices.

Blindly wasting resources, such as memory, really is foolish, but in this case it's not an engineering tradeoff, it's an example of a programmer who is too far from their machine trying to “just make it work.” That's not programming, that's just typing. Software engineers and programmers worth their expensive chairs and high salaries know that they don't want to waste resources, so they try to figure out what the best- and worst-case scenarios are and how they will affect the other possible users of the system. Users in most cases are now just other programs, rather than other people, but we all

know what happens to a system when it starts to swap things out of memory onto secondary storage. That's right, your dev-ops people call you screaming at 3 a.m. That's fine if you're asleep but not if you're drunk, or maybe it's the other way around. Actually, screaming people are never that much fun, except at a concert.

You mention that this software is for a "high-performance" device, and if by that you mean it goes in a typical 64-bit server-class machine, then no one is really going to notice a megabyte, or four, or even eight. A high-end server-class machine is unlikely to have less than four gigabytes of RAM. Even if you allocate four megabytes at system startup time, that's one-tenth of one percent of the available RAM. People writing in Java will suck down far more than that just starting their threads. Are you really going to worry about less than a tenth of a percent of memory?

If you had told me that this driver was for some limited-memory-size embedded device, I would give other advice, since that system might not have 4 GB of RAM; but then again, given what most phones and tablets have in them now, it might.

People are often right when they say, "Waste not, want not," but it's also important to take your moderation in moderation.

KV

Dear KV,

I'm converting a Web application to run on a mobile platform. While the application doesn't handle banking information or anything with crazy security like that, it does still require the user to type a password. Our password requirements aren't too strict, though we do have a minimum size of eight characters and require one uppercase letter and one nonalphanumeric character. Because on-screen typing is so inaccurate, our product development folks want us to relax our password requirements even more, allowing the user to have a four-character, all-lowercase password, which they call a mobile PIN code. The mobile PIN would work only from the mobile app and not on the Web. I've tried to explain to them that four characters simply aren't enough, but maybe if we're restricting this to the mobile app, it will be OK. What do you think?

Pinned Down

Dear Pinned,

I was wondering when someone would write a letter like this. Having typed on a variety of tablets in the past few years, I knew it was only a matter of time before some marketing or product-type person would ask an engineer to dumb down security for convenience. I'm glad you point out that your application doesn't relate to banking, as I would have had to write back immediately and ask, "Which bank?!", and that could only lead to trouble.

As a quick aside, I do find it interesting that the world thinks only of banks when they think of security. While it would be very bad for someone to transfer all the money out of your bank account into their own, the fact of the matter is that a lot of really bad things can happen online that don't relate directly to cash flow. Weak passwords can leave users open to identity theft, stalkers, kidnappers, and their exes. It's up to you to think about which of those is better or worse than losing some cash. I would take losing money over being stalked by my exes.

The problem with a four-character password—as you point out—is that it's too short and easy to guess. A PIN with a card, such as a card used at an ATM, is employed because it requires physical possession of something, the card, to be effective. Schemes for protecting users online, such as

those involving passwords, depend on the user presenting a combination of something they have, something they are, or something they know. A password is an example of the latter. The schemes can be mixed and matched, such as the PIN and a physical card, where the user has something and knows something.

The problem with current mobile devices is that they are much more limited in their ability to handle user input than, say, a computer with a keyboard. On-screen keyboards aren't very good, which is probably why someone at Google thought to use a pattern for device unlocking on the Android. I find that system a bit silly and easy to shoulder surf, but it's good to see someone trying to do something differently.

It would be nice to pretend that the device itself would be proof of something the user has, but since the point of the password is to prevent a malicious party from accessing the user's data after the device has been stolen or, more likely, left in a bar, it needs to be sufficiently strong to deter an attacker, even if that attacker is also drunk. If you're unable to fight back on password complexity, it's time to break out the lockout option. A four-digit pin code is mostly a problem if you allow the attacker a large number of attempts to guess the PIN. If, after three tries, you lock out the user for five minutes, and then let the user try again, it's going to take a long time for the attacker to try enough PINs to guess the right one—that is, if the user hasn't picked a common PIN, such as 1234 or 2580 (an exercise for readers is figuring out why that second code is so common; for a more academic study of the problem of PINs see the paper, "A birthday present every eleven wallets? The security of customer-chosen banking PINs," by Joseph Bonneau et al. from the Computer Laboratory of the University of Cambridge at http://www.jbonneau.com/doc/BPA12-FC-banking_pin_security.pdf).

Being the bastard that I am, I also like the idea of three failed tries causing the device not to work at all, including erasing all local data and then requiring the user to go through a recovery flow that does not involve the device. If your application is used by people who are often mentally impaired in some way (and no, I don't mean that they're upper management, but I do mean when it is used for social networking), then an annoying recovery system is not going to get past your product development folks. Too many people want to upload pictures of their friends in compromising positions, and that, alas, requires compromising on security.

KV

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

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.

© 2013 ACM 1542-7730/13/0800 \$10.00