Pointless PKI

As prickly as he is, Kode Vicious actually enjoys reading your comments and might even respond to them in print. But it's your programming questions that are the true lifeblood of the column. So don't be shy, shoot off your latest humdinger to kv@acmqueue.com. If we publish it you'll receive not only the benefit of KV's sage advice, but also a very cool *Queue* coffee mug.

Dear KV,

I work at a large Web-based company, and we're looking for a way to secure our traffic. Unlike most companies, this is not to secure the traffic between remote offices but actually to secure all the traffic inside the company, between the front-end Web servers and our back-end databases. We've had problems in the past with internal compromises, and management has decided that the only way to protect the information is to encrypt it during transmission. We won't be storing the data encrypted because it's too hard to get everyone to rewrite their applications. Building a system like this is no easy feat because we have thousands of servers involved in making our systems work. I'm building a PKI system to handle all the keys necessary—one for each service we provide—and am wondering if you have any advice on how to secure data inside a company, as opposed to making the service itself secure.

Keyed up over Security

Dear Keyed,

Yes, you're right, building such a system is no easy feat, and the worst part is, even if you succeed it will be completely pointless. Several things in your letter confuse me, so I'll try to address them logically, which is more than I can say for your management, which seems to be in what

Got a question for Kode Vicious? E-mail him at kv@acmqueue.com—if you dare! And if your letter appears in print, he may even send you a Queue coffee mug, if he's in the mood. And oh yeah, we edit letters for content, style, and for your own good!

A koder with attitude, KV ANSWERS

YOUR QUESTIONS.

MISS MANNERS HE AIN'T.

is kindly referred to as "reactive mode" and what I would term, "putting your head in your sand" (or somewhere else).

By "internal compromises," I suspect you mean that some employees have been making off with your data. Internal compromises and leaks are a risk at any company—and the larger the company, the bigger the risk. The more people you have involved, the more likely you're going to wind up with a few people whom you should not have hired. How do you keep these infernal internal people from doing things with the data that they should not? I can tell you that encrypting all traffic is not going to help much. It is unlikely that an inside attacker would put a packet sniffer on the network, collect a day's or week's worth of data, and then walk out with it. Sifting through that much information is far too much work, and besides, you've given them a much easier target.

If your company is storing data in a back-end database, then that is where internal attackers will go to get their data. Why sift through packet traces when a few SQL statements and a DVD burner or fast connection would provide you with much better data? If you're storing sensitive data, then it is the data that needs to be secured, not the network! What if the attacker walks off with the backups, as has happened in several cases recently? If the data in the database is not secured—that is, hashed or encrypted—then he who has the backups has the data.

Another concern that most people don't understand in this sort of system is the concept of "need to know." Governments and the military, which are just two sides of the same coin, attempt to set up systems such that sensitive data is seen or modified only by people who actually need to work with that data, hence, "need to know." Databases and other computer systems can also be set up in similar ways, such that only the small number of people who need to work with any particular bits of data actually work on it. Honest people won't care that they don't have access to all the data because they have what they need to do their jobs, and the dishonest ones will have access to less data, thereby reducing the chance of a compromise.

A frequent mistake people make in setting up secure systems is to encrypt everything. If you encrypt all data, then everyone has to have keys and the keys can be lost or stolen, thereby leading to a compromise. Encrypt only what you must encrypt to secure your business; then only a small number of people will need keys or access to the sensitive data.

Finally, you don't say anything about auditing in your letter. The best way to find the dishonest people in your system is not by encrypting all the communication but by auditing when sensitive data is read, modified, or deleted. Keeping a log of "who did what to whom," and reviewing that log on a regular basis, is the best way to find the abusers of your system.

I'm sorry I didn't tell you how to implement a good PKI system. It's a fascinating topic, but it's not one that is going to help you at all, except in making your corporate masters feel more secure, when in reality they won't be.

KV

Dear KV,

I read the comments on heisenbugs ("Kode Vicious Bugs Out," April 2006) and am surprised you did not mention the approach that tends to work the best in solving these particular bugs. That approach is the random passerby who looks over your shoulder and immediately points out the error.

Passing

Dear Passing,

The debugging practice you're referring to is what I call the "Stupid Programmer Trick." There are actually two different versions. The one you mention is actually the less reliable of the two, because it depends on a chance encounter.

The other version, which I prefer, is where I walk over to a coworker—it could be anyone, not just an engineer, just someone to stand there and go "uh huh" a lot—and start explaining the problem I'm having. It could even be a marketing person, but then I have to buy them drinks and that cuts into my own drinks budget.

To use this trick, you start explaining your problem, pointing at the code or diagrams you're using to think about the problem. If you're talking to someone with a clue, you might get lucky and that person might find the bug, or at least ask you a good question. But at some point, BANG, you smack your forehead—being bald I do this a lot; it makes a nice slap sound—and say, "Eureka!" and jump from the bath. No, wait, that was someone else.

At any rate, you get that lovely feeling of having found the problem. Thanks for reminding me.

KV

Dear KV,

I read your response to Hungry Reader (April 2006), and I indeed have all of the books you listed except Raj Jain's (*The Art of Computer Systems Performance Analysis*, Wiley, 1991), which I am going to get. I would include just one more that I feel is indispensable: *The Mythical Man-Month* by Frederick P. Brooks (Addison-Wesley, 1975; republished 1995). It's good to get a copy for your manager, too. What do you think? Should this one be on your list?

Nostalgic over PDP-11s

Dear Nostalgic,

At the risk of drawing another Web comment on being an antique for recommending an older book, yes, your suggestion is good. The thrust of my original response was technical books, whereas *Mythical Man-Month* is a management book, although one that gives me dry heaves a lot less than most books in the management section of my local bookstore.

I remember the book fondly for several reasons. Its conclusions were obvious to anyone who had worked in a software company for more than five minutes, yet you can still use it to beat stupid managers over the head with. It has stood the test of time for two reasons: One is that it was well written, an uncommon quality; the other reason is that people haven't gotten any smarter about managing large projects in the last 50 years. One other good thing about *Mythical Man-Month* was that it was short enough that I could read it and return it in time to recoup the full price from my university bookstore. I took the money and bought beer for a Fourth of July party. No, I am not kidding. So, I don't have my copy anymore, but I do have some fond, if confused, memories.

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 has made San Francisco his home since 1990.

© 2006 ACM 1542-7730/06/0700 \$5.00