Kode Vicious Plays in Traffic

**WITH INCREASING
COMPLEXITY
COMES
INCREASING RISK.**

Dear KV,

I hear that many cars today are built as distributed systems containing hundreds of CPUs that control the smallest bits of the car. These components, with millions of lines of code in them, seem to be very complicated—more so than a typical operating system. This doesn't sound like a terribly great idea, given that today we struggle to understand multicore behavior of systems in the presence of optimizing compilers, let alone the challenges posed by distributed systems that have no access to atomic operations. I hear that they're even planning to use Ethernet moving forward. I'm scared that a car might malfunction or get exploited and run me over. What can we do to remedy this situation?

Sincerely,

A Frightened Citizen Running from Cars

Dear Frightened,

The only thing we have to fear is fear itself—and poorly written code that has kinetic side effects and off-by-one errors. In other words, we have much to fear. There is a very simple answer to all this car silliness, and it is, of course, bicycles. Nice, mechanical, muscle-driven machines with nary a processor anywhere near them.

Unfortunately, it is unlikely that bicycles will replace automobiles anytime soon, and as you point out,

automobiles are becoming increasingly automated. As people who work in software, we know that this is a terrible idea because we see how much terrible code gets written and then foisted upon the world. At one point, KV might have suggested that more stringent requirements, such as those used in the aerospace industry, might have been one way to ameliorate the dangers of software in the four-wheeled killing machines all around us, but then Boeing 737s started falling out of the air and that idea went out the window as well.

There is no single answer to the question of how to apply software to systems that can, literally, kill us, but there are models to follow that may help ameliorate the risk. The risks involved in these systems come from three major areas: marketing, accounting, and management. It is not that it is impossible to engineer such systems safely, but the history of automated systems shows us that it is difficult to do so cheaply and quickly. The old adage, “Fast, cheap, or correct, choose two,” really should be “Choose correct, and forget about fast or cheap.” But the third leg of the stool here, management, never goes for that.

There is a wealth of literature on safety-critical systems, much of which points in the same direction: toward simplicity. With increasing complexity comes increasing risk, in part because humans—and I’m told that management is made up of humans—are quite bad at understanding both complexity and risk. Understanding the safety parameters of a system means understanding the system as a whole, and a simpler system is easier to understand than a complex one.

Think of what happens when the network traffic showing your kids their favorite TV programs interferes with the signal from the brake pedal.

The first design principle of any safety-critical system must be simplicity. Systems such as Ethernet, which you reference in your letter, are known to be complex, with many hidden failure modes, and so it is a poor choice for use in a safety-critical system. But I hear accounting screaming about the cost of extra wiring in the harness of the car's control system. "Think how much money we can save if all the signals go over a single pair of wires instead of a harness with 10!" In response, we must say, "Think of what happens when the network traffic showing your kids their favorite TV programs interferes with the signal from the brake pedal."

Which brings up the next design principle, separation of concerns. A safety-critical system must never be mixed with a system that is not safety critical, for this both increases complexity and lowers the level of safety provided by the system overall. The brakes and steering are far more important than the entertainment system, at least if you think that stopping in time for a light is more important than singing along to "Life on Mars." This type of design failure has already shown up in a variety of systems that are safety critical, including automobiles.

A third, but by no means final, design principle can be stated as, "Don't talk to strangers." Many of the latest features in systems—such as automobile systems—are meant to make them an extension of the Internet. I don't know if you've seen the Internet lately, but it is not a safe space. Why anyone should be distracted by email—or gods help them, Slack—while driving is beyond KV's understanding, but this is something marketing clearly wants to push, so, against all sanity, it is definitely

Related articles

DNS Complexity

Although it contains just a few simple rules, DNS has grown into an enormously complex system.

Paul Vixie

<https://queue.acm.org/detail.cfm?id=1242499>

Tom's Top Ten Things Executives

Should Know About Software
Software acumen is the new norm.

Thomas A. Limoncelli

<https://queue.acm.org/detail.cfm?id=3325792>

Programming Languages

Kode Vicious

<https://queue.acm.org/detail.cfm?id=1035604>

happening. There have already been spectacular takeovers of cars in the field by white-hatted attackers, so it ought to be obvious that this is an important design principle, and anyone who tells me this problem can be solved with a firewall will be dragged behind an SUV and dumped off a cliff. Adding a firewall adds complexity, violating tenet #1 above.

The irony of this list is that it's not new. These principles have existed for at least 40 years in one form or another, but now they have more weight, or perhaps kinetic energy.

KV

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.

Copyright © 2020 held by owner/author. Publication rights licensed to ACM.