What is a CSO good for?

**SECURITY
REQUIRES
MORE THAN AN
OFF-THE-SHELF
SOLUTION.**

Dear KV,

The little startup I'm working for must be getting bigger because we just hired someone to be our "chief security officer," which I place in quotes because I'm not quite sure what that actually means. Most of the developers I work with seem to write good code, which, if I understand some of your previous columns, means that we should also have relatively good security.

What confuses me about the CSO is that whenever our chief architect—my boss—tries to talk to him about how our systems function, I get the feeling that the CSO doesn't listen. In fact, much of what the CSO has done since coming on hasn't focused on the security of our software. Instead, he buys third-party security products and then pushes them on the development groups and the rest of the company. Often these systems get in the way of getting work done, and from time to time they just fail, which means we either stop using them or find ways to bypass them.

Is this normal? I like working at startups, and this is the first time I've been at one that has gotten big enough to hire such a person, so maybe this is just how big companies work and it's time to move to yet another startup, where security is part of our work rather than something that's bought for us.

Bought and Paid For

Dear Bought,

Asking “What is a CSO good for?” is like asking “What is any executive good for?” This is a topic that is probably too meaty for me to address in a single column, but let’s see if I can at least partially answer your question here. CSOs are like snowflakes; no two are alike. Actually, the snowflake theory of any group is total bull; there are definitely distinct categories you find in any role, whether it’s a developer, marketer, or C-level executive. Like any executive, a CSO is supposed to be a leader with a concentration in security, someone who can: (1) survey and understand the threats against the company on many levels; (2) describe those threats to various groups within the organization; and then (3) develop plans to protect the company, its people, and its assets against those threats.

The CSO is *not* a security engineer, so let’s contrast the two jobs to create a picture of what we should and should not see.

The CSO thinks about (actually a good one has nightmares about) various security threats and then ranks them in various orders. One possible ordering is based on the likelihood of the threat being realistically carried out. Another ordering is based on the downside risk of the threat actually coming to fruition. A good example is an attack on a single system versus one that takes out a whole set of systems.

Imagine you are building an app that runs on someone’s phone, a very common job. There is some nonzero probability that someone will attack the app. The downside risks of a successful attack on a single instance of the app (say, where the attacker can get at some data but has to

Good security,
like good underwear,
is not
one size fits all.

have physical possession of the person's phone) versus the one where they can remotely get data from many, or all, instances of the app are very different. In the former case, you have screwed over one customer, and in the latter, you have screwed over your entire user base. These mental calculations, writ large, are what a CSO spends time thinking about.

A security engineer, on the other hand, builds systems such as software, network architectures, or other artifacts that implement a particular security feature against an identified threat. Using the same threat-model map, a security engineer works to prevent a successful attack on the system.

The case of the phone application remains illustrative. A security engineer will work on the application code to make sure that it stores any data that must remain secret—for example, keys used to carry out secure network communications—in a secure place such as a TPM (Trusted Platform Module), a hardware security module that is commonly provided in modern, mobile hardware. Of course, the security engineer knows why this is necessary, but is not going to simultaneously worry about how the company's network routers are protected from attack.

Once CSOs have developed a threat-model map, they have to figure out if it's correct and applies to the systems being developed. Good security, like good underwear, is not one size fits all. A thong and boxer briefs do not provide the same level of protection, and while there are definitely occasions for both, they are rarely the same occasion. The fact that you think your CSO is not listening to your chief architect should give you pause. I would actually

expect that their discussions would be quite intense, and I've worked at one startup where no such conversation was carried out without a lot of yelling. If CSOs do not understand what they're trying to help protect, how can they protect it?

This brings me to one of the least understood parts of security work, both by its practitioners and by those upon whom it is practiced. The security role is always a helping role: that person, or, more often, group of people, must be there to help everyone around them understand the threats and be able to point them to resources that help them solve their problems.

Too much of the security industry is full of people with military backgrounds or military frames of mind, where one can command and compel people to act in certain ways under harsh penalties. Most software companies are not military units, and most engineers laugh at this type of command and control. You pointed out that you and your colleagues have started to work against the security systems being foisted upon you, and this is actually the worst possible outcome, because it makes systems far less secure than if the security system wasn't put in place at all.

The other issue you described, the CSO's penchant for buying systems of sometimes dubious quality, has gotten worse with the spread of the Internet and the need to secure more and more systems. Before the Internet, you had to secure only your computer, the hulking thing in the basement, and a few dialup modems against insiders, which was bad enough. Now, your systems and software can be attacked from anywhere and everywhere, and if you look at your SSH (Secure Shell) logs, you'll see that they are.

Related articles

➡ Pointless PKI

A koder with attitude, KV answers your questions. Miss Manners he ain't.

<https://queue.acm.org/detail.cfm?id=1147526>

➡ Browser Security: Appearances Can Be Deceiving

A discussion with Jeremiah Grossman, Ben Livshits, Rebecca Bace, and George Neville-Neil

<https://queue.acm.org/detail.cfm?id=2399757>

➡ CTO Roundtable: Malware Defense

The battle is bigger than most of us realize.

<https://queue.acm.org/detail.cfm?id=1731902>

As any industry grows, it inevitably draws a percentage of people and companies who are there “just to make a buck,” and that makes careful and deliberate decision making even more important. There is plenty of FUD sown by the security industry, which you can see in their advertising in pretty much any airport: *Spammers are out to get you and there are two viruses in every laptop!* There is definitely a nasty threat landscape, and though there continues to be interesting work in mitigations, countermeasures, and overall

development practices, security will remain an arms race, at least for the foreseeable future.

What your CSO is currently practicing is called “checkbook security,” a particularly dangerous way to deal with threats. While there are definitely good security products on the market, the fact is that without a careful plan and careful deliberation, you can't simply achieve security by buying a product or a suite of products. You have to think about how to use the product, if it addresses an identified threat, and if it integrates with your company's work. A failing in any of these three areas means you're just pissing good money down a drain—money that could be better spent on drink and drugs to ameliorate

those threat-landscape nightmares.

KV

Code 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. Neville-Neil is the co-author with Marshall Kirk McKusick and Robert N. M. Watson of *The Design and Implementation of the FreeBSD Operating System* (second edition). He is an avid bicyclist and traveler who currently lives in New York City.

Copyright © 2019 held by owner/author. Publication rights licensed to ACM.