Communications of the ACM, Vol 60, No. 11, Pages 8-9

Bring 'Diseased' Software Up to Code

We agree with Vinton G. Cerf's advice in his Cerf's Up column "Take Two Aspirin and Call Me in the Morning" (Sept. 2017) that we all practice better "cyber-hygiene" but must quarrel with the continued use of public health as a metaphor for cyber security. If we as computing professionals intend to improve the cybersecurity of our critical infrastructures, rather than merely tolerate their current "diseased" state, we must think differently. We thus propose a return to an older metaphor for software, likening its structures to physical structures and its architecture to the architecture of physical buildings. Such thinking suggests we consider how to build software that will not fall over when attacked or build it from weak materials unable to bear expected stress.

Software we rely on for critical functions (such as controlling medical devices, delivering electrical power to households, and guiding automobiles) must conform to an appropriate set of constraints, just as physical structures conform to building codes before they can be occupied. A third party must be able to certify conformance to these constraints, just as building inspectors certify buildings.

These codes are best developed by those who build the systems, not by government, though governments might use them once they are in place. An industry-consensus building code, with third-party assessment of conformance, can help the marketplace reward those who build systems with fewer vulnerabilities.

Over the past few years, with support from the IEEE's Cybersecurity Initiative and the National Science Foundation, workshops have been held to begin to develop such building codes for medical-device software and for power-system software.[1,2] In addition to these draft codes, related promising developments include *Consumers Reports'* collaboration with the Cyber Independent Testing Laboratory (http://cyber-itl.org/) to develop methods for publicly rating software products, and UL's (http://www.ul.com) development and use of a standard for certifying cybersecurity assurance of products.

Treating software security as a public health problem is not likely to lead past the decades-old ideas of aftermarket vaccines, antivirus, and quarantine. Providing evidence that software is at least free of specified classes of vulnerabilities covered by an appropriate building code can yield a more effective market incentive for companies to produce the cyberinfrastructures we all need—and that are up to code.

Robert K. Cunningham, Lexington, MA,
Tom Haigh, Minneapolis, MN,
Carl Landwehr, New Buffalo, MI, and
Alfonso Valdes, Urbana, IL

References

1. Haigh, T. and Landwehr, C. *A Building Code for Medical Device Software Security.* Technical Report. IEEE Computer Society, Mar. 2015; https://www.computer.org/cms/CYBSI/docs/BCMDSS.pdf
2. Landwehr, C.E. and Valdes, A. *Building Code for Power System Software Security.* Technical Report. IEEE Computer Society, Mar. 2017; https://www.computer.org/cms/CYBSI/docs/BCPSSS.pdf

Author Responds:

It is always a pleasure to hear from Carl Landwehr with whom I have had a long acquaintance and for whom I have great respect. An interesting challenge with his building/architecture metaphor relates to the way software is often constructed these days by incorporating (vast) libraries of code reflecting, perhaps, uncertain provenance. There is also the uncertainty of software interactions across the network that may never have been tested until a chance encounter leads to a breach. None of this invalidates the building-code metaphor but might make it more difficult to establish that the ensemble meets the desired code standards and properties. I am, in fact, very interested in the development of programming aids that will do a much better job of assessing source code against desirable properties of attack resistance and identifying potential sources of weakness. Until we have reliable ways of producing the kind of code Landwehr et al. and I likely agree we need and want, we may also want to argue that infected or vulnerable devices ought not to be naively tolerated and that owners (and suppliers) bear at least some responsibility for observing diligent software hygiene.

Vinton G. Cerf, Mountain View, CA