



# "What We Do Now, Kemosabe?"

Guest Author **Vinton G. Cerf**, Google

*Perhaps now is the time to start a campaign to apply more formal methods to software evaluation, especially if they can be automated for analysis and testing.*

**A**s a child growing up, one of my favorite television programs was *The Lone Ranger*. When I came down with COVID-19 in March 2020, I was so debilitated that I ended up watching at least 50 (maybe more) episodes of that show on YouTube. Inevitably, the plots reached a place where it was not clear how to resolve the unresolvable, at which point Tonto, the Lone Ranger's American Indian companion, would invariably ask, "What we do now, Kemosabe?"


It is true that that some researchers claim that "Kemosabe" is a mispronunciation of the Spanish "quien no sabe,"

which means "who does not know," that is, "you dummy." (The same researchers may note that "Tonto" in Spanish means "fool" and that the dynamic duo were calling each other names for nearly a decade! "Kemosabe" may also mean "faithful friend" in the Potawatomi language. Maybe they are just made-up names and mean nothing in particular.) Nevertheless, we might ask ourselves

the same question about the state of software security in both online and offline contexts. We have managed to become deeply dependent on digital infrastructure, which is largely based on billions of lines of software written in scores of languages. There are open source libraries that are heavily used to create or support new applications. It is sometimes suggested that because the source code is open for scrutiny by anyone, bugs that can be exploited by hackers are absent because anyone can spot and fix them.

When everyone is in charge, no one is in charge. Indeed, it is likely that a significant portion of open source software harbors exploitable bugs. In some cases, these vulnerabilities may not be discovered for years or even decades.<sup>1</sup>

This observation suggests to me that we might well invest in the systematic scrutiny of open source software libraries, ranking the analysis by the popularity (that is, use) of the code found there. Many companies that rely on software offer “bounty” programs that pay for bugs found and reported. Of course, some players keep the bugs private for purposes of exploitation, suggesting that any bug bounty program should encourage widespread reporting efforts. More generally, however, it seems that we need to invest in better programming environments to expose mistakes before software is actually released into use. A programming environment that can model the software and facilitate model checking to expose errors might be helpful. These things exist, for example, Estelle<sup>2</sup> or TLA+,<sup>3</sup> but Estelle, TLA+, and other

such verification tools are less widely applied than one would like. Given our evident and increasing dependence on massive amounts of software, much of which is interacting in complex ways on the Internet, perhaps now is the time to start a campaign for the application of more formal methods to software evaluation, especially if they can be automated to produce abstractions whose properties can be analyzed and tested. 

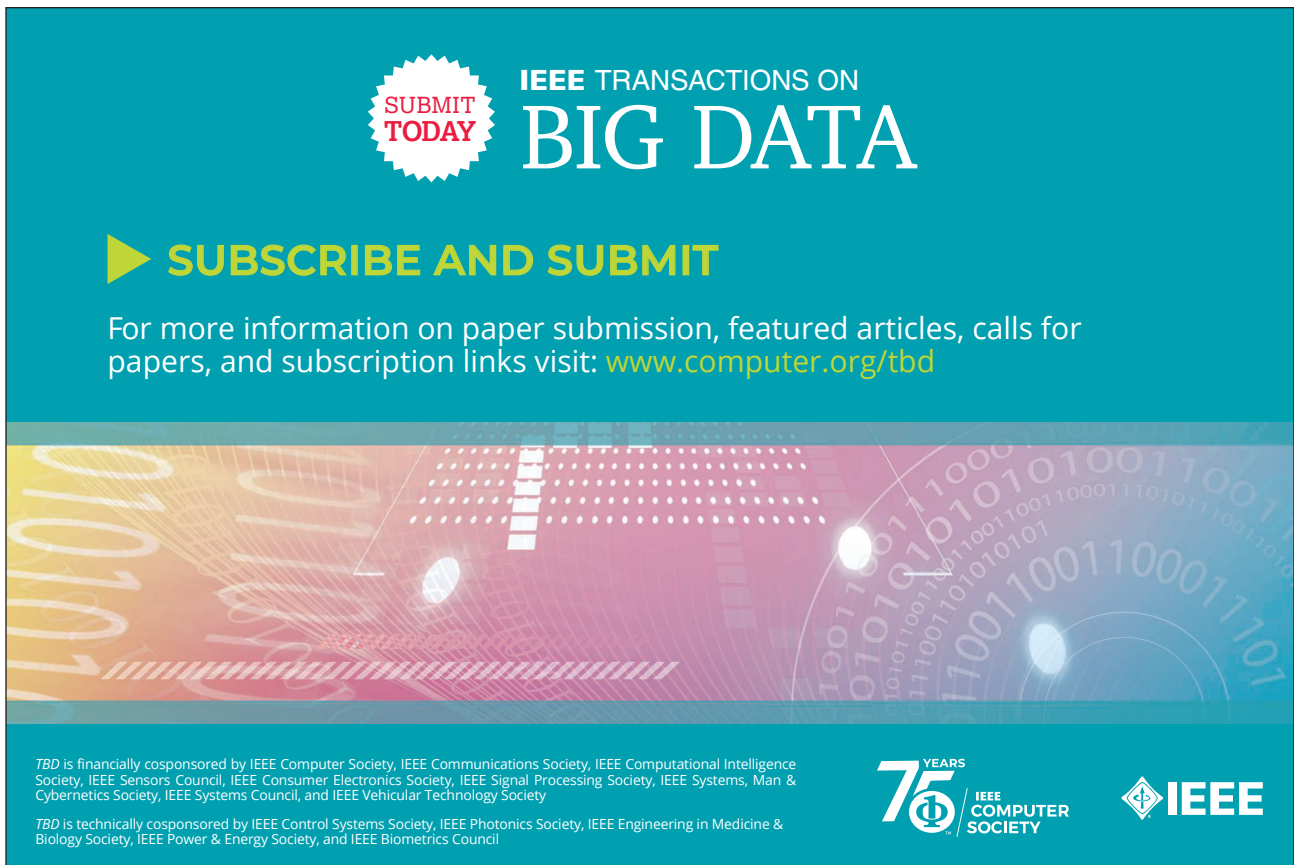
REFERENCES

1. P. Johnson. “11 software bugs that took way too long to meet their maker.” CSO Online. [www.csoonline.com/article/3404334/11-software-bugs-that-took-way-too-long-to-meet-their-maker.html](http://www.csoonline.com/article/3404334/11-software-bugs-that-took-way-too-long-to-meet-their-maker.html) (accessed Jan. 19, 2021).
2. S. Budkowski and P. Dembinski, “An introduction to Estelle: A specification language for distributed

systems,” *Comput. Netw.*, vol. 14, no. 1, pp. 3–23, 1987. [Online]. Available: [www.semanticscholar.org/paper/An-Introduction-to-Estelle%3A-A-Specification-for-Budkowski-Dembinski/b7ba8821259b1da17bfed1effebd32d7b376de1a](http://www.semanticscholar.org/paper/An-Introduction-to-Estelle%3A-A-Specification-for-Budkowski-Dembinski/b7ba8821259b1da17bfed1effebd32d7b376de1a). doi: 10.1016/0169-7552(87)90084-5.

3. K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz, “A TLA+ proof system,” in *Proc. Combined KEAPP - IWIL Workshops*, 2008, pp. 17–37. [Online]. Available: <https://lampport.azurewebsites.net/pubs/keappa08-web.pdf>

VINTON G. CERF is the vice president and chief Internet evangelist at Google, McLean, Virginia, 22102, USA. Contact him at [vint@google.com](mailto:vint@google.com).



**SUBMIT TODAY** IEEE TRANSACTIONS ON **BIG DATA**

**▶ SUBSCRIBE AND SUBMIT**

For more information on paper submission, featured articles, calls for papers, and subscription links visit: [www.computer.org/tbd](http://www.computer.org/tbd)

TBD is financially cosponsored by IEEE Computer Society, IEEE Communications Society, IEEE Computational Intelligence Society, IEEE Sensors Council, IEEE Consumer Electronics Society, IEEE Signal Processing Society, IEEE Systems, Man & Cybernetics Society, IEEE Systems Council, and IEEE Vehicular Technology Society

TBD is technically cosponsored by IEEE Control Systems Society, IEEE Photonics Society, IEEE Engineering in Medicine & Biology Society, IEEE Power & Energy Society, and IEEE Biometrics Council

**75 YEARS**  
IEEE COMPUTER SOCIETY 