DOI:10.1145/2736281

Dorothy E. Denning

Privacy and Security
Toward More Secure Software

Two proposals intended to reduce flaws in software use two very different approaches for software security.

Last year, the National Institute Standards and Technology (NIST) added 7,937 vulnerabilities to the National Vulnerability Database (NVD), up from 5,174 in 2013. That is approximately 22 per day, or almost one every hour. Of these, 1,912 (24%) were labeled "high severity" and 7,243 (91%) "high" or "medium."[7] Simply put, they cannot be ignored.

As I read reports of new vulnerabilities and the risks they enable, I wonder whether it will ever end. Will our software products ever be sufficiently secure that reports such as these are few and far between? Or, will they only become more prevalent as more and more software enters the market, and more dangerous as software increasingly controls network-enabled medical devices and other products that perform life-critical functions?

In this column, I will explore two proposals aimed at reducing software flaws. The first, which involves the U.S. government cornering the vulnerability market, I believe, could make the problem worse. However, the second, which involves holding companies liable for faulty software, is an idea whose time has come.

Cornering the Vulnerability Market

Many software companies, including Microsoft, Google, and Mozilla, operate bug bounty programs, paying security researchers who bring new security flaws to their attention. Other companies serve as brokers, buying vulnerabilities and exploits from security researchers, and then selling or donating them to product vendors and other customers. To the extent the end consumers in this growing market are the companies whose products are flawed, the market serves to strengthen software security. But when end consumers are intelligence agencies and criminals who use the information to exploit target systems, the vulnerability market exposes us all to greater risk.

To further reduce software vulnerabilities beyond what the market has achieved so far, we could look for ways that encourage the pursuit of vulnerabilities with the goal of getting them fixed, but discourage their sale and use for target exploitation. One possibility, suggested by Dan Geer, is for the U.S. government (USG) to openly corner the vulnerability market.[6] In particular, the USG would buy all vulnerabilities and share them with vendors and the public, offering to pay say 10 times as much as any competitor. Geer argues this strategy will enlarge the talent pool of vulnerability finders, while also devaluing the vulnerabilities themselves. Assuming the supply of vulnerabilities is relatively sparse, the approach could eventually lead to a situation where most vulnerabilities have been exposed and fixed, rendering any cyber weapons that exploited them useless. In addition, since researchers finding new zero-day will maximize their earnings by selling them to the USG, fewer zero-days should end up in the hands of adversaries.

The cost of Geer's proposal seems reasonable. Current prices for vulnerabilities range from a few hundred to several hundred thousand dollars. If we consider the approximately 8,000 vulnerabilities added to the NVD in 2014 and assume an average price of $1,000, then the cost of purchas-

ing these vulnerabilities would be $8 million, a drop in the bucket for the USG. Even if the average price rose to $100,000, the annual cost would still be reasonable at $800 million. However, the costs could become much higher and the problems worse if the program perversely incentivized the creation of bugs (for example, an inside developer colluding with an outside bounty collector).[1] Costs could also rise from outrageous monetary demands or the effects of more people looking for bugs in more products.

I especially worry that by shifting the cost from the private sector to the USG, companies would lose an economic incentive to produce more secure software in the first place. As it is, an empirical study by UC Berkeley researchers of the bug bounty programs offered by Google and Mozilla for their respective browsers, Chrome and Firefox, found the programs were economically efficient and cost effective compared to hiring full-time security researchers to hunt down the flaws.[5] Would it not be better to shift the incentives so it was more economical to invest in secure software development than in patching flaws in deployed code that puts users at risk?

Another issue is whether the USG would be willing to disclose all vulnerabilities. Under current policy, software flaws it uncovers are generally to be disclosed to vendors in order that they can be patched, but they can be kept secret and exploited when there is "a clear national security or law enforcement" reason.[10] At the same time, it seems likely that many discovered vulnerabilities will never reach the USG, being held for the purposes of exploitation by criminals and foreign governments. And many persons might simply oppose reporting them to the USG.

Finally, vulnerability disclosure has the downside of increasing the risks of those using the reported products, at least until they can acquire and install the necessary patches. Consider ShellShock, a flaw in the UNIX Bourne-again shell (Bash), which lets attackers remotely execute code or escalate privileges. Disclosure of the flaw allowed attackers to harvest vulnerable computers into a botnet that sent out over 100,000 phishing email messages.[3] A Symantec study found that attacks exploiting particular zero-day vulnerabilities increased by as much as 100,000-fold after their disclosure.[2]

Software Liability

A better approach to reducing vulnerabilities would be to hold software companies liable for damages incurred by cyber-attacks that exploit security flaws in their code. Right now, companies are not liable, protected by their licensing agreements. No other industry enjoys such dispensation. The manufacturers of automobiles, appliances, and other products can be sued for faulty products that lead to death and injury.

In *Geekonomics*, David Rice makes a strong case that industry incentives to produce secure software are inadequate under current market forces, and that one way of shifting this would be to hold companies accountable for faulty products.[8] Geer proposes that software companies be liable for damages caused by commercial software when used normally, but that developers could avoid liability by delivering their software with "complete and buildable source code and a license that allows disabling any functional-

ity or code the licensee decides."[6] The escape clause, which would cover free and open source software, would allow users to inspect and cut out any software they did not trust.

My main concern with Geer's proposal relates to absolving any code offered commercially from liability. As a practical matter, very few users are in a position to inspect source code. Even those that are can miss significant flaws, as seen with Heartbleed, a flaw in OpenSSL that gives attackers access to sensitive information, and also ShellShock. In addition, exemption does nothing to incentivize the production of more secure open source code. At the same time, penalizing a large, volunteer community for flaws in their code would be both difficult and distasteful.

A better way around this dilemma might be to exempt the immediate developers of open source code, but hold accountable any company that embeds it in their products or that offers services for open source products. Under such a provision, if an individual or group of volunteers offers a free, open source App, they would not be accountable, though any company offering it through their App store would be.

This compromise would incentivize software companies to pay greater attention to security in all of the code they offer through their products and services, regardless of whether the code is developed in house, by a contractor, or within the open source community; and regardless of whether the product is released with source code or the capability to disable certain functions. Moreover, given that many companies contribute to open source development, they would be incentivized to promote secure coding practices in the open source community as well as within their own development teams.

Importantly, as with other forms of liability, software liability should be tied to standards and best practices, as well as the damage and harm that result from security flaws. The objective is not to penalize companies who invest considerable resources in software security but find their code vulnerable to a new exploit that nobody had anticipated. Rather, it is to bring all software up to a higher level of security by punishing those who are negligent in this domain, for example, by putting out code that fails to check inputs. Standards and best practices for secure coding have advanced considerably, and readers interested in learning more might start with CERT's Secure Coding Web portal.[4]

Developing a suitable liability regime will be a challenge, as the system must address the concerns of both software developers and users.

The argument is often made that software liability will inhibit innovation, but we should inhibit the introduction of faulty software. Moreover, assigning liability will likely stimulate innovation relating to secure software development. Another argument against software liability is that it will raise the price of software. While this may be true, it should lower the costs we all pay from cyber-attacks that exploit software vulnerabilities, costs that have been rising over the years and fall on the backs of users as well as software companies.

Conclusion

Bug bounties emerged under current market forces and are likely here to stay. I oppose a program that would attempt to have the U.S. government corner and fund this market, in part because it would reduce the incentive for software companies to produce more secure code and could make the problem worse.

A better strategy is one that increases the incentives for the development of secure software but decreases those for putting out sloppy code. One way of achieving this is by holding companies responsible for all the code they sell and service, including both proprietary and open source. Under this strategy, companies could be sued for damages caused by cyber-attacks that exploited flaws in their code, and penalties would be inflicted according to whether the code was developed under standards and best practices for secure coding.

Because software licenses and the Uniform Commercial Code severely limit vendors from liability for security flaws in their code, companies today cannot be effectively sued or punished when they are negligent and the flaws are exploited to cause economic harm.[9] Legislation or regulation is needed to change this and remove the ability of companies to exempt themselves through licensing agreements. Developing a suitable liability regime will be a challenge, however, as the system must address the concerns of both software developers and users. Perhaps a good start would be for ACM to sponsor a workshop that brings together a diverse community of stakeholders and domain experts to recommend a course of action.

Of course, holding software companies accountable will not solve all our security woes. Many cyber-attacks exploit weaknesses unrelated to faulty software, such as weak and default passwords and failure to encrypt sensitive information. But companies are liable when their systems are attacked, and they can be successfully sued for not following security standards and best practices. The time has come to make software vendors liable as well. C

References

1. Baker, H. Re: Zero-day bounties. *The Risks Digest 28.25* (Sept. 9, 2014).
2. Bilge, L. and Dumitras, T. An empirical study of zero-day attacks in the real world. *CCS'12* (Oct. 16–18, Raleigh, N.C.); http://users.ece.cmu.edu/~tdumitra/public_documents/bilge12_zero_day.pdf.
3. Carman, A. Shellshock used to amass botnet and execute phishing campaign. *SC Magazine* (Oct. 15, 2014); http://bit.ly/1Df7Slg.
4. CERT Coding Standards; http://www.cert.org/secure-coding/index.cfm.
5. Finifter, M., Akhawe, D., and Wagner, D. An empirical study of vulnerability rewards programs. *USENIX Security 13*; https://www.eecs.berkeley.edu/~daw/papers/vrp-use13.pdf.
6. Geer, D. Cybersecurity as realpolitik. Blackhat 2014; http://geer.tinho.net/geer.blackhat.6viii14.txt.
7. National Vulnerability Database, National Institute of Standards and Technology; https://web.nvd.nist.gov.
8. Rice, D. *Geekonomics: The Real Cost of Insecure Software*, Addison Wesley, 2008.
9. Scott, M.D. Tort liability for vendors of insecure software: Has the time finally come? *Maryland Law Review 67*, 2 (2008); http://digitalcommons.law.umaryland.edu/cgi/viewcontent.cgi?article=3320&context=mlr.
10. Zetter, K. Obama: NSA must reveal bugs like Heartbleed, unless they help the NSA. *Wired* (Apr. 15, 2014); http://www.wired.com/2014/04/obama-zero-day/.

Dorothy E. Denning (dedennin@nps.edu) is Distinguished Professor of Defense Analysis at the Naval Postgraduate School in Monterey, CA.

The views expressed here are those of the author and do not reflect the official policy or position of the U.S. Department of Defense or the U.S. federal government.