CSIA RFI Responses, January 15, 2019

Update to the 2016 Federal Cybersecurity Research and Development Strategic Plan RFI Responses

DISCLAIMER: The RFI public responses received and posted do not represent the views and/or opinions of the U.S. Government, NSTC Subcommittee on Networking and Information Technology Research and Development (NITRD), NITRD National Coordination Office, and/or any other Federal agencies and/or government entities. We bear no responsibility for the accuracy, legality or content of all external links included in this document.

RFI Response: Federal Cybersecurity R&D Strategic Plan
Evidence-based Secure Development Practices

Dr. Sam Weber
Carnegie Mellon University CyLab
samweber@acm.org

January 15, 2019

Thank you for the opportunity to provide input to the Federal Cybersecurity R&D Strategic Plan. This submission is a response to question 4, "What challenges or objectives not included in the 2016 Strategic Plan should be strategic priorities for federally funded R&D in cybersecurity?"

This submission argues that "Evidence-based Secure Development Processes" should be a strategic priority and a critical dependency in the upcoming strategic plan.

Over the decades there has been enormous progress in cybersecurity technology. Ultimately, though, the nation's cybersecurity posture rests upon practitioners correctly designing, implementing and maintaining their systems. Unfortunately, while cybersecurity challenges have been mounting, knowledge about secure development processes has not been keeping pace.

For example, Acar et al [1] have shown that existing cryptographic libraries are incredibly difficult for practitioners to use correctly. Despite the fact that all of their experimental participants were experienced programmers, 20% of the functionally-correct code that they produced and thought was secure was, in fact, not. One implication of this is that a high percentage of government-funded cryptographic technology is unable to be successfully deployed.

As another illustration of this problem, Wang et al [5] studied the mechanisms provided by Facebook and Microsoft to allow single-sign-on on third-party applications. (For example, web pages that allow users to sign-on using their Facebook ids.) They discovered that the majority of real-world, deployed, applications used these mechanisms incorrectly, causing security vulnerabilities – 67% of web pages using Facebook sign-on did so improperly! Therefore the problem is not limited to cryptographic libraries, but a general problem in secure development that even major companies have difficulty handling.

Traditionally the assumption has been that "bugs are bugs": security bugs are like other kinds of bugs and will be addressed by improvements in software engineering techniques. This assumption has been disproven: Morrison et al [4] shows that vulnerabilities are discovered later in the development cycle and are more likely to be resolved by changes in conditional logic than non-security defects. Camilo et al [2] found that there was only a weak correlation between non-security bugs and vulnerabilities and that characteristics such as source lines of code and number of features were more related to security bugs. They also found that files

with the highest non-security defect density did not intersect with the files with the highest security vulnerability density. Zaman et al [6] showed that security bugs are involve more developers and impact more project files than non-security bugs and have more complex fixes.

The conclusion is that security engineering is not the same as regular development: security raises issues that aren't well-addressed by traditional development practices. Our current state of knowledge concerning security engineering is depressingly but accurately summarized by Danezis:

> In security engineering we have quite a few case reports, particularly relating to specific failures, in the form of design flaws and implementation bugs. We also have a set of methodologies as well as techniques and tools that are meant to help with security engineering. Which work, and at what cost? How do they compare with each other? What are the non-security risks (cost, complexity, training, planning) associated with them? There is remarkably little evidence, besides at best expert opinion, at worse flaming, to decide. This is particularly surprising, since a number of very skilled people have spent considerable time advocating for their favorite engineering paradigms in the name of security: static analysis, penetration testing, code reviews, strong typing, security testing, secure design and implementation methodologies, verification, pair-coding, use of specific frameworks, etc. However, besides opinion it is hard to find much evidence of how well these work in reducing security problems. [3]

What is needed, then, is *evidence-based* secure-development methodologies. Not only should practitioners be able to adopt scientifically-validated development practices, but the reliance on evidence allows the community to successively improve and refine said practices.

Furthermore, such work is a critical dependency: the inability of practitioners to effectively design, implement and maintain secure systems is a continuous inhibitor. Furthermore, the innovations fostered by the other portions of the Strategic Plan require appropriate development methodologies to be realized in actual systems.

Again, thank you for the opportunity to provide this feedback.

References

[1] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek, and C. Stransky. Comparing the Usability of Cryptographic APIs. *IEEE Symposium on Security And Privacy*, 2017.

[2] F. Camilo, A. Meneely, and M. Nagappan. Do Bugs Foreshadow Vulnerabilities? A Study of the Chromium Project. *The 12th Working Conference on Mining Software Repositories*, 2015.

[3] G. Danezis. Security engineering: What works? `https://conspicuouschatter.wordpress.com/2014/12/18/security-engineering-what-works/`, 2014.

[4] P. J. Morrison, R. Pandita, X. Xiao, R. Chillarege, and L. Williams. Are vulnerabilities discovered and resolved like other defects? *Empirical Software Engineering*, 23(3):1383–1421, 2018.

[5] R. Wang, Y. Zhou, S. Chen, S. Qadeer, D. Evans, and Y. Gurevich. Explicating SDKs: Uncovering Assumptions Underlying Secure Authentication and Authorization. In *USENIX Security*, pages 399–414, 2013.

[6] S. Zaman, B. Adams, and A. Hassan. Security versus performance bugs: A case study on firefox. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011.