Editorial Preface

Software Security Problems Never Go Out of Style

Martin Gilje Jaatun, SINTEF Digital, Trondheim / University of Stavanger, Stavanger, Norway

Whereas we have to accept that routinely creating "ordinary" software that is 100% secure will probably never happen, the field of software security has progressed to the point where there is a lot of knowledge readily available for those who care to look. I have previously mentioned the BSIMM (Williams et al., 2018) and OpenSAMM (Chandra, 2008) initiatives, that enumerate a large number of software development practices that contribute toward creating more secure software. All practices may not be a good fit for all organizations, but I believe that everyone would benefit from taking the time to run through the list of BSIMM or OpenSAMM practices (which one you choose may be more of a personal preference), noting the ones you perform and those you don't. Then, assessing the latter category, making a conscious decision on whether these activities really are irrelevant (or deprecated for whatever reason). The BSIMM report may help in this respect, as it can tell you which activities are more common in similar industries as yourself – this still does not mean that whatever other people do necessarily are right for you, but it should be an incentive to think extra carefully about these activities.

The Open Web Application Security Project (OWASP) maintains a list of the Top Ten vulnerabilities in web applications (Søhoel et al., 2018), and though recently updated, the changes in the last decade are much less dramatic than one might expect. The top vulnerability is still injection, despite the wide coverage it has received in the academic and popular press. Stack overflow was described at length over 20 years ago (Aleph One, 1996), but we see that buffers, heaps and stacks are overflowing in freshly minted software. This seems to confirm that there is a need to radically change how we train the next generation of developers and software engineers, and I would go so far as to postulate that no student should be allowed to learn how to program without at the same time learning how to program securely!

This issue contains three articles. First, Chaim and Cruzes present a survey of existing techniques to detect buffer overflow vulnerabilities in "What do we know about buffer overflow detection? A survey on techniques to detect a persistent vulnerability", confirming that this is still a highly relevant problem. Then, Lescisin and Mahmoud explore how secure software can be developed by the means of dynamic analysis tools in "Evaluation of Dynamic Analysis Tools for Software Security", appropriately addressing the two topics of memory safety (as in buffer overflow) and input validation (as in injection). Finally, Osis and Nazaruka address what they consider to be the weakness of software engineering in their contribution "Theory Driven Modeling as the Core of Software Development", where they argue that that software development based on mathematical formalism combined with the principle of architectural separation of concerns is a way forward.

Martin Gilje Jaatun
Editor-in-Chief
IJSSSP

REFERENCES

Aleph One. (1996). Smashing the stack for fun and profit. *Phrack, 49*(7).

Chandra, P. (2008). *Software assurance maturity model*. Retrieved from https://www.opensamm.org/

Søhoel, H., Jaatun, M. G., & Boyd, C. (2018). OWASP Top 10 - Do Startups Care? *Paper presented at the 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Glasgow.

Williams, L., McGraw, G., & Migues, S. (2018). Engineering Security Vulnerability Prevention, Detection, and Response. *IEEE Software*, *35*(5), 76–80. doi:10.1109/MS.2018.290110854