End of Module Assignment:

E-Portfolio and Reflection Submission

Secure Software Development

Word count: 953

E-Portfolio:

https://busilas.github.io/eportfolio/module3.html

Reflection

This paper draws on the learning and the change in knowledge, skills, and personal development experienced, assessing the Secure Software Development (SSD) module's contribution to how I see secure software engineering. Employing the framework by Rolfe et al. (2001), I will use the "What? So What? Now What?" structure to organise my reflections.

What?

The module trained me on the ideas, and techniques of securing software development that range from, aspects of software development, such as the Agile paradigm, risk-oriented design, secure coding standard, and how secure coding can be applied, tested, and deployed. In a nascent stage, I observed, by doing so, the benefit of leaving behind conventional methods, especially by adopting Agile instead of a rigid methodology, and the flexibility of methods in adapting to problems such as security flaws (Sharma & Bawa, 2020). Flowcharts based on the Unified Modelling Language

(UML) activity were particularly effective at visualising what could go wrong and structuring the mitigations (Fowler, 2004). The work on UML diagrams assisted the process of structuring secure design, however, it also helped to ensure good communication with team members.

During the module practical exercises facilitated the application of theory. For example, the design of a command-line interface (CLI) and security features such as role-based access control demonstrated that security concepts can be implemented in software in vivo (Morton, 2022). During this hands-on experience, the difference between what is theoretically known and what is practiced in the real world was overcome particularly well. Additionally, the acquisition of the industry's best practices, including ISO/IEC 27000 and OWASP wisecracks, provided a strong starting point from which to identify and secure vulnerabilities (Hjerppe et al., 2019; OWASP, 2021).

The collective assignment allowed us to face alternative conceptions of secure development and improve teamwork and communicative abilities. Exploring real-world security attack scenarios such as TrueCrypt (Junestam & Guigo, 2014) brought out the import of continuous relevance in terms of technological evolution and the need to acquire new knowledge within the field of human factors involved in security intrusions (Zhang et al., 2019). In addition, the identification of tools including Python linters and other automated testing frameworks enhanced developers' capability to develop secure and maintainable code (Kapil, 2019).

So What?

The knowledge obtained during the module has changed fundamentally my understanding of software development. Before this module, my focus was on functionality and performance. However, I realize now the paramount importance of the need to implement security early in the development life cycle (SDLC) (Martin

Kung, 2018). Security is a fundamental prerequisite to be considered from the design stage and even up to the time of deployment. For instance, I underestimated risks associated with bad coding hygiene, e.g., lack of input validation, which the module illustrated both through practice coding exercises and bides (Ponta et al., 2020). Having an awareness of the OWASP Top Ten vulnerabilities has also given me a clearer view of possible traps at the development stage.

One of the most outstanding learnings was understanding that software security can only be achieved by being able to understand the role played by an organisational culture. The risk-sensitive setting guarantees that security (Choperena et al., 2019) compliance is not limited to the lowest rung of the ladder to C-suite managers. This idea found resonance with me during a joint exercise, during which we examined security failures due to cultural failures, e.g., insufficient training or lack of responsibility. Cognitive positive attitudes toward security are of equal importance to the technology that protects.

A further important lesson was the direct use of cryptographic libraries to protect private data. Encrypting a CLI application enabled a deep insight into the efficacy and inadequacy of cryptography. For instance, though, if the encryption algorithms are sufficiently powerful, their capability decreases if the keys are weak or are leaked (Kedziora et al., 2017). In trial and error, I observed that key rotation and secure storage have become essential and frequently painless if best practices are implemented by design as part of a plan.

The module further emphasised the reciprocity of technical and social skills. My team discussions enhanced my communication capacity, my capacity to find common ground, and my capacity to garner the consensus of all my team members. For example, this learning showed that flexibility and empathy are essential to developing good teamwork.

Now What?

In terms of the future, and building on what has been learned in the module, I intend to apply it to future projects. To promote sustainable progress, I will exploit my experience with powerful coding specifications, e.g., applied to material closely related to the notes prepared by OWASP and to ISO/IEC standards.

On a broader level, gaining knowledge during the SSD module will encourage me to foster a security-first mindset for whatever project I will be involved in by sharing expertise and advocating for industry standards such as regular code reviews and vulnerability assessments. I hope to help the community reduce security concerns (Meriah & Rabai, 2019). This module has also ignited my interest in SSD research areas, such as Machine Learning for threat detection and cryptographic engineering.

Conclusion

The SSD module has been a game changer for my growth, as it has given me the powers and mindset to preserve security throughout the software system design lifecycle. I have gained in-depth knowledge of secure software techniques through theoretical studies, learning by doing, and teamwork. Furthermore, I have learned through experience and practice, to be an active, reflective learner and problem-solver, both of which can be valuable in future study and working life. The skills and information gained during this module are not only just academic successes but serve as the foundation for continual improvement and lifetime learning in the dynamic field of SSD.

Reference list

Choperena, A., Hernández-Lloreda, M.J., & López-González, M.D. (2019). ‘Actionable insights from reflective practices in software engineering’. Journal of Software Engineering, 25(3), pp. 215-230.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modelling Language. 3rd edn. Addison-Wesley.

Hjerppe, R., Virtanen, S., & Korpela, M. (2019). ‘Ensuring GDPR compliance in software development projects’. Information Systems Journal, 29(2), pp. 110-129.

Holtzblatt, K., & Marsden, N. (2022). Agile and Beyond: Leveraging Iterative Development to Enhance Security. Cambridge: MIT Press.

Junestam, A., & Guigo, N. (2014). Open Crypto Audit Project: TrueCrypt Security Assessment. iSEC Partners.

Kapil, P. (2019). ‘Python linters for secure and high-quality coding’. Python Journal of Coding Practices, 18(4), pp. 42-56.

Kedziora, D., Kotowski, K., & Baranowski, M. (2017). ‘A comparative analysis of cryptographic libraries: TrueCrypt vs. VeraCrypt’. Cybersecurity Review, 15(3), pp. 28-36.

Martin, J., & Kung, D. (2018). Security by Design in Modern Software Development. 2nd edn. CRC Press.

Meriah, S., & Rabai, L.B.A. (2019). ‘ISO/IEC 27000 compliance for secure software engineering’. International Journal of Information Security, 18(2), pp. 85-100.

Morton, J. (2022). ‘Role-based access control in CLI applications’. Journal of Secure Computing, 30(2), pp. 76-89.

OWASP (2021). ‘OWASP Top Ten Security Risks’. Available at: https://owasp.org (Accessed: 10 January 2025).

Ponta, S.E., Plate, H., & Sabetta, A. (2020). ‘Automated vulnerability scanning for open-source libraries’. Empirical Software Engineering, 25(5), pp. 3176-3199.

Rolfe, G., Freshwater, D., & Jasper, M. (2001). Critical Reflection for Nursing and the Helping Professions: A User’s Guide. Basingstoke: Palgrave Macmillan.

Sharma, A., & Bawa, R.K. (2020). ‘Identification and integration of security activities for secure agile development’. International Journal of Information Technology, 12(1), pp. 1-15.

Zhang, Y., Li, X., & Luo, Y. (2019). ‘Understanding vulnerabilities in TrueCrypt: Lessons for secure software development’. Journal of Software Security Practices, 14(4), pp. 98-113.