Collaborative Discussion – UML Flow Charts

Summary

The introductory three weeks introduced students to various ways to develop secure software during agile processes. Several engineering process definitions exist to aid the integration of security processes into an agile methodology, such as Cigatel Security Touchpoints, Common Criteria, and Microsoft SDL of CLASP (Comprehensive, Lightweight Application Security Process) (Baca & Carlsson, 2011). Of the engineering processes researched, the Microsoft SDL is for incorporation into existing development processes.

The importance of risk awareness in an organisation is of utmost importance because cybersecurity threats are external to organisations and users who utilise or abuse existing systems or software. ISO/IEC Standard 27000 defines multiple terms related to cybersecurity threats, risks, and vulnerabilities to understand the range of risks involved. Also, Hadlington (2017) notes that an organisation's security is often subservient to users' primary requirements. Excessive security negatively impacts users' ability to complete tasks. As a result, users may (usually) end up circumventing established security procedures. These users could end up unintentionally causing organisational harm or increasing the chances of security exploits (Mazarolo & Jurcut, 2019).

UML is perfectly suitable to model software security vulnerabilities—as described by the OWASP Top Ten. Two particular UML diagram types are simple behavioural models that depict a flow of activities along a given path and "swim lanes". Swim lanes are particularly useful for clarifying asset boundaries and attack surfaces that require security. UML diagrams may also highlight the likelihood of a cybersecurity weakness, access control vulnerability, need for data governance policy, potential risk or exposure. However, secure software still requires monitoring, sufficient logging, and continual improvements to reduce security events.

Several best practices—adherence to coding guidelines, validating inputs, using in-built language facilities, following the principle of least privilege (CISA, 2007)—support secure software development. Another consideration is for developers to consider security risks and vulnerabilities when using open source libraries (O'Reilly, 2018; Koussa, 2021). Lastly, secure software development requires buy-in from management, and a framework such as the Secure Software Development Framework may greatly assist organisations (NIST, 2021) to achieve that gioal.

References

Baca, D. & Carlsson, B., (2011) Agile development with security engineering activities. Proceedings of the 2011 International Conference on Software and Systems Process:149-158.

CISA (2017) Cybersecurity & Infrastructure Security Agency. Available from https://us-cert.cisa.gov/bsi/articles/knowledge/principles/least-privilege [Accessed 1 Sep 2021]

Hadlington, L. (2017) Human factors in cybersecurity; examining the link between Internet addiction, impulsivity, attitudes towards cybersecurity, and risky cybersecurity behaviours. Heliyon, 3(7), p.e00346.
DOI: **https://doi.org/10.1016/j.heliyon.2017.e00346**

Koussa, S. (2021) 13 tools for checking the security risk of open-source dependencies. Available from https://techbeacon.com/app-dev-testing/13-tools-checking-security-risk-open-source-dependencies [Accessed 1 Sep 2021]

Mazzarolo, G. & Jurcut, A.D. (2019). Insider threats in Cyber Security: The enemy within the gates. arXiv: https://arxiv.org/ftp/arxiv/papers/1911/1911.09575.pdf [Accessed 1 Sep 2021]

NIST (2021) Computer Security Resource Center: Secure Software Development Framework. Available from https://csrc.nist.gov/Projects/ssdf [Accessed 1 Sep 2021]

O'Reilly (2018) Mitigating known security risks in open source libraries: Fixing vulnerable open source packages. Available from https://www.oreilly.com/content/mitigating-known-security-risks-in-open-source-libraries/ [Accessed 1 Sep 2021]