# Collaborative Discussion – TrueCrypt

## Summary

Considering the abrupt termination of a software tool such as TrueCrypt due to various security vulnerabilities, it presents a strong imperative for software development teams to exercise careful consideration over their product's security concerns. Such security concerns include software and hardware, and a single failure in one oftentimes has catastrophic consequences in the other.

To facilitate the reduction of software security vulnerabilities, using a Secure Development Lifecycle (SDLC) is highly recommended. For instance, the Microsoft SDL (Sharm & Bawa, 2020) recommend familiar steps for software teams who have exposure to Microsoft tools, while the Common Criteria ISO process (Baca & Carlsson, 2011) may be a suitable fit for general security-focused development processes. Another aspect raised to help better identify security vulnerabilities is for teams to focus on using the latest libraries, adherence to accepted design patterns and engaging in regular code reviews. However, Paul et al. (2021) not that for code reviews to be effective, more time must be spent on reviews, more reviewers must actively participate, and coders must focus on reducing cyclomatic complexity. An increased focus on security vulnerabilities may lead organisations to "security fatigue" (Shanton et al., 2016) which arises when users are bombarded constantly about threats to their data and cause uses to consider the threat-benefit risk and rewards.

Therefore, it is imperative for organisations to balance security policies with customer needs (GDPR concerns) and system requirements. The focus must be placed on the developers of systems, ensuring they write low vulnerability code using best practices and regularly and often address security-focused program concerns raised by customers. Their use of selected cryptographic algorithms should not be based on the ease of use but rather on the sensitivity of the data that requires security control. Such a concept is applicable to hardware and software, and for this, the use of a TPM module (Microsoft, 2021) can serve as a valuable hardware-based solution to enabling more secure software solutions, but, Muñoz & Fernandez (2020) note the use of TPM

requires enhanced levels of security expertise and that a TPM cannot substitute for poorly implemented software flows.

# 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.

Microsoft (2021) Trusted Platform Module Technology Overview. Available from: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-overview [Accessed 11 October 2021]

Muñoz, A. & Fernandez, E.B. (2020). TPM, a pattern for an architecture for trusted computing. Proceedings of the European Conference on Pattern Languages of Programs, 2020:1-8.

Paul, R., Turzo, A.K. & Bosu, A. (2021). Why security defects go unnoticed during code reviews? a case-control study of the chromium os project. IEEE/ACM 43rd International Conference on Software Engineering (ICSE):1373-1385.

Sharma, A. & Bawa, R.K. (2020) Identification and integration of security activities for secure agile development. International Journal of Information Technology: 1-14.

Stanton, B., Theofanos, M.F., Prettyman, S.S. & Furman, S. (2016). Security fatigue. It Professional, 18(5):26-32.