Secure Software Testing Methodologies: Adopting Code Review and Penetration Testing to Promote Threat Defense

Sean Carroll

Villanova University

"This paper or presentation is my own work. Any assistance I received in its preparation is acknowledged within the paper or presentation, in accordance with academic practice. If I used data, ideas, words, diagrams, pictures, or other information from any source, I have cited the sources fully and completely in footnotes and bibliography entries. This includes sources that I have quoted or paraphrased. Furthermore, I certify that this paper or presentation was prepared by me specifically for this class and has not been submitted, in whole or in part, to any other class in this University or elsewhere, or used for any purpose other than satisfying the requirements of this class, except that I am allowed to submit the paper or presentation to a professional publication, peer reviewed journal, or professional conference. In adding my name following the word 'Signature', I intend that this certification will have the same authority and authenticity as a document executed with my hand-written signature.
Signature Sean Carroll"

Abstract

This paper explores current best practices in secure software testing methodologies. It is in the effort to find the most optimal approach for the consideration and advising of industry. Our ultimate purpose seeks to ensure protection and mitigation of advanced persistent threats (APTs) and cybersecurity asymmetric threats by promoting a gradual methodology that follows the secure software development lifecycle (SSDLC) within the industry environments that seek improvement. The importance of designing a secure software testing methodology is one that is often underemphasized in the secure software development lifecycle due to current methodologies of design and procedural processes. The security quality assurance throughout the lifecycle is often bothersome and conflicting to certain phases of the development lifecycle—specifically the initial stages of design and implementation. Software security touchpoints must not be seen as a separate phases of secure software design implemented post-development, but must be integrated into each phase of the development lifecycle. With consideration to current secure software testing methodologies, this paper discusses code review and penetration testing along each phase of the SSDLC. If frameworks and procedures are defined to mitigate miscommunication and align current approaches with proposed practices, an optimal methodology rife with efficiency can be realized.

Secure Software Testing Methodologies: Adopting Code Review and Penetration Testing to Promote Threat Defense

Integration of standard security software touchpoints along the lifecycle of secure software has clear and defined benefits that are indisputable. Testing of secure software faces a challenge of quantifying its foreseen benefits and importance within software development, despite ensuring the objectives in gradual security quality assurance. With the recent asymmetric cyberattacks posing a threat to private and public sectors, the importance of secure software testing and developing a secure framework is highlighted. Code review and penetration testing are practices that can optimize current secure software testing methodologies upon proper adaptation and integration within environments of interest.

Literature Review

Batcheller, Fowler, Cunningham, Doyle, Jaeger, and Lindqvist (2017) suggests that software often meets functional requirements to meet user demands, however, the consequences of not considering security requirements alongside other necessary requirements creates an environment where software is vulnerable. Consequently, insecure software has unexpected consequences that impact not only the end-user, but the long-term buildup of insecure software mechanisms created by a poor security protection approach can spell unexpected losses and immediate losses to the commercial entity of concern (Batcheller et al., 2017). The 2017 research conducted by Batcheller et al. suggest from large-scale cyberattacks and the devastation that there is an inherent flaw in the traditional software development lifecycle due to the fact that security measures are taken in the testing phase; however, the optimal approach is to develop security protections and testing practices that are present along each phase of the development lifecycle—especially early phases such as design. The previously reference work asserts through

previously conducted research that the testing phase in the secure software development lifecycle results in inefficient use of funds and the possibility for issues to initially be discovered only upon reaching the operational stage.

Macleod, Greiler, Storey, Bird, and Czerwonka (2017) expand upon previous analysis conducted on the challenges and best practices of code review. The research conducted in Macleod et al.'s 2017 article analyzed a large-scale study of Microsoft code review methodologies and consisted of qualitative analysis from targeted focus groups and quantitative analysis through surveys that attempted to mitigate bias. The mentioned research highlights the many challenges encountered when implementing ideal frameworks from academia with industry groups. Macleod et al. (2017) asserted the realistic tradeoffs that are often faced when developers are faced with varied requirements and time restraints. It is clear that the amount of code review needed to seek optimal levels of security quality assurance are specific to every environment.

Shaukat, K., Faisal, A., Masood, R., Usman, A., and Shaukat, U. (2017) suggest that there are many varied approaches and testing methodologies that can implement the tools and practices of penetrating testing. Shaukat et al.'s 2017 paper declares that it is of importance to separate the procedures of threat identification and patching of such vulnerabilities to optimized the efficiency in which the penetration process is conducted.

Discussion

Macleod et al.'s research illustrates the challenges of code review in practice. It appears that the overarching problem involves the time efficiency of communication and response between code reviewers and code developers. The motivation and main objective of code review is for code improvement, which ultimately promotes security quality assurance of the software. It

is important to highlight that spending too much time on code review can be detrimental to the overall objective of the development team of concern. Furthermore, the article broadens this idea of seeking balance between tradeoffs by presenting Parkinson's law of triviality, which is often given the term bikeshedding by industry and is important to consider in the entire testing process to reach optimal practices for testing. Shaukat et al.'s discussion into the different methodologies of testing in penetration is of particular interest. The use of whitebox, greybox, or blackbox testing, where the various levels of access provided by users is varied is important to consider when desiring to detect different levels of vulnerabilities in a system based on user privileges.

Conclusion

Time efficiency of development and the importance of guiding developers in a way that meets main objectives must be considered when conducting code review and penetration testing. The integration of testing into every phase of the secure software development lifecycle necessitates that management seeks efficiency in communication. Emphasizing the importance of best software testing procedures to parties that hold influence on software of concern is necessary to optimize efficiency and promote progress. It appears that an optimal secure software testing methodology adopts beneficial frameworks and effectively communicates optimal procedures and best practices to parties of concern. In this manner, the objectives and long-term security goals are integrated throughout all phases of the software development lifecycle.

References

Batcheller, A., Fowler, S. C., Cunningham, R., Doyle, D., Jaeger, T., & Lindqvist, U. (2017). Building on the Success of Building Security In. IEEE Security & Privacy, 15(4), 85–87.

MacLeod, L., Greiler, M., Storey, M.-A., Bird, C., & Czerwonka, J. (2017). Code Reviewing in the Trenches: Understanding Challenges and Best Practices. IEEE Software, 1–1. https://doi.org/10.1109/MS.2017.265100500

Shaukat, K., Faisal, A., Masood, R., Usman, A., & Shaukat, U. (2016). Security quality assurance through penetration testing. In Multi-Topic Conference (INMIC), 2016 19th International (pp. 1–6). IEEE.