Journal VIII: Portfolio Reflection

Barbara A. Kelly

Southern New Hampshire University

Security should not be left until the end of a software development life cycle, because it is the most important aspect of the project. Security should always be a top-most priority and implemented at the beginning of the development life cycle. As with all aspects of coding, there are standards to adhere to, and learning to apply secure coding standards should be of absolute importance.

I think a core set of high-level secure software development standards should be added to a software development lifecycle. Clear communication of this structure should be addressed and shared with all stakeholders and developers at the onset of a project. This will help to reduce the number of vulnerabilities in released software, help to mitigate the potential impact of exploitation of undetected vulnerabilities, and to prevent future incidents.

 When thinking about Zero Trust, trust is a vulnerability in terms of security.  Once a layer on the network is breached, users and attackers are free to move laterally and access data with no limit. it's important to identify and know how traffic moves across the organization in relation to whatever layer users have access to. Understanding who the users are, which applications they are using and how they are connecting is the only way to enforce policy that ensures secure access to the data.  The potential of breach without zero trust is too easy - it's almost a misstep in overall secure coding.

One way to prevent threats, is to validate user input, as this can prevent SQL injection attacks. This can be done by using strong encryption mechanisms to protect the keys, encoding data so that special characters are suitably handled, and ensuring that the server validates all input. Another method is to implement access control, which involves limiting permissions to the least team members as possible. It is recommended that users should not be able to rollback their privileges, and that user’s activities cannot be cached – to keep sensitive information protected. Another important method is to recognize input length, as this can prevent buffer overflow attacks. This should be done by only accepting a set input length and data types within certain fields. Insecure coding by not instilling a secure database design, can lead attackers to have easy access to data that can be manipulated. This can cause malicious attacks done through the main database. This one goes along with validating user input data and ensuring that write access is only given to authorized team members.

I think I would ensure that secure coding methods are in place – and specifically that of encryption. If encryption is enabled at the onset of the code and functions, this can help to build a solid base for security. I like the idea of encrypting passwords using the XOR algorithm. The XOR cipher makes sense to me when it is enabled for simple strings by text and key. I would also be sure to prevent buffer overflow, by limiting input length of appropriate fields where necessary. By being proactive with just these two methods, I feel, would keep a project on track with security, and not downplaying security attacks

References

Dhillon, G., Ph.D. (2015). WHAT TO DO BEFORE AND AFTER A CYBERSECURITY BREACH? Retrieved April 06, 2021, from <https://www.american.edu/kogod/research/cybergov/upload/what-to-do.pdf>

SecureIca. (2020, July 20). Importance of secure coding with it's best practices. Retrieved April 05, 2021, from <https://medium.com/infosec/importance-of-secure-coding-with-its-best-practices-b43d4c26bd0b>