In the modern era of technology, creating secure software plays a vital role in creating a safe digital environment. When a code we created isn't secure enough, it will have devastating impacts on individuals, organizations, and society as a whole. Even though we are very well aware of it, why aren't we creating secure coding? What makes it so hard? In this blog, we will walk through some of the main challenges in secure coding practices and discuss the ways to mitigate the risks that arise when we don't follow secure coding practices.

When practicing secure coding the fundamental challenge is that the computer is unintelligent, despite their awesome capabilities. They follow instructions blindly but correctly without understanding the situation and the purpose of the command or instructions. So, attackers can use it to exploit vulnerabilities and flaws in the codes.

Below are the security issues and the ways to mitigate the risks and practice secure coding standards.

Validate input – using data from unknown and untrustworthy websites would lead to vulnerabilities in the software we create. Always be cautious and suspicious about external data from network interfaces, command line arguments, and user-controlled files. So always make sure to validate data from external sources which can mitigate vast amounts of vulnerabilities.

KISS – (Keep It Simple, Stupid), always try to keep the code as simple as possible. When trying to do complex programming, the chance of making an error is high when implementing and configuring the code. even if we managed to give a security to it, it also becomes complex along with the code making it hard for developers to secure it.

Principles of least privilege – every process should run with the least privileges required to do the task. Any higher privilege should be used for as long as is necessary to finish the privileged job. This method mitigates the chances of an attacker running arbitrary code with enhanced privileges.

Auditing and logging – make sure to frequently log into the software created to monitor it after it was deployed in the production environment, so any potential incidents can be found and fixed in the beginning stage to mitigate huge loss of data in the later stage.

Deep defense practice – developers should always use multiple layers of security in the program so that if one falls down for an attack, there are still several more to defend the system and program from being exploited by their vulnerabilities. For instance, using secure programming methods in conjunction with secure runtime environments could mitigate the possibility of vulnerabilities in the code still present at the time of deployment being exploited in the operational environment.

Define security requirements – When security requirements are not defined, the security of the resulting system cannot be evaluated effectively. therefore, it is important to identify and document security requirements early in the development life cycle and to ensure that subsequent development artifacts are evaluated for compliance with those requirements.

Given that computers are unintelligent and the constantly changing world of cyber threats, writing safe software is obviously difficult. However, developers may dramatically lower the risks related to software vulnerabilities by employing secure coding methods, carrying out frequent security audits, and promoting a culture of security awareness. Writing secure software is crucial in the current digital era; it's not simply a work; it's a duty that may help safeguard people, businesses, and society from the escalating dangers of the internet.

References

Top 10 Secure Coding Practices - CERT Secure Coding - Confluence. (n.d.). https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices

Snyk. (2020). Secure coding practices every developer should know. Snyk. https://snyk.io/learn/secure-coding-practices/