Unit 1 Collaborative Discussion

Software security is, perhaps, the most important aspect of the software development lifecycle (SDLC). Sensitive and personal data is shared across many systems such as smart devices, laptops, phones. The development of the internet of things (IoT) is only increasing the sharing of personal data, placing more importance on digital security. Pillai defines security as the:

'ability of a system to avoid damage to its data and logic from unauthenticated access, while continuing to provide services to other systems and roles that are properly authenticated.'(Pillai, 2017)

The Open Web Application Security Project (OWAP) is a nonprofit foundation that works to improve the security of software. OWAP produce guidelines, standards and principles for software developers to adhere to. OWAP identify broken access control as the number one security risk to web applications (OWASP, 2021). Broken access control occurs when the system allows a user to act outside of their usual permissions. For example, this could occur when:

- Incorrect permissions have been allocated to a role
- Elevation of privilege within a module of the software
- Insufficient control checks being executed within the software when modifying data

This can be mitigated through secure coding strategies. An example of a secure coding strategy would be the principle of least privilege where every process in the system is executed with the minimum system privilege to achieve functionality. Additionally, access to system code and data should be protected through effective authorisation defined by the software architecture.

At the beginning of the SDLC, the system architecture should define the security requirements using the CIA aspects of security architecture:

- Confidentiality – data is not exposed to unauthorised access or modification
- Integrity – data is free from external manipulation
- Availability – data is accessible to authorised users

This should be defined for the various roles of the system stakeholders: user, admin, etc.

Of course, a well-defined system architecture and the implementation of secure coding strategies may not be sufficient to project the system from the evolving threats and attacks. Testing and threat modelling are necessary parts of the SDLC which enable review and revision where appropriate. There are 2 main development models to consider: waterfall and agile.

The traditional waterfall model places an emphasis on design before coding. While it could be argued that the time dedicated to development of the architecture would mitigate risk, the reality is that as coding and testing take place near the end of the cycle any emerging threats are a risk. Furthermore, this linear approach means it is inherently challenging to amend and adapt which exposes the model to more risk. This can be mitigated through a spiral approach which adapts the waterfall model to spiral through the engineering and evaluation phases, making it more secure.

The agile approach is a contrast to the waterfall model where software is developed through a series of iterative developments. This means that the process is adaptable and it is easy to amend code to respond to testing and security threats. However, the speed of the process (often described as sprints) can in itself be a risk which has led to the development of secure scrum which emphasises security as well as task completion.

References

Pillai, A.B. (2017) Software Architecture with Python. Birmingham, UK. Packt Publishing Ltd.

OWASP. (2021) OWASP Top 10:2021 AO1:2021 – Broken Access Control. Available from: https://owasp.org/Top10/A01_2021-Broken_Access_Control/ [Accessed 5th August 2024]