Software Security Measures

Authors: Kaylen Eastwood 2210194

T-yondre Leslie 2210176

Iyana Taylor 22010251

UNIVERSITY OF TECHNOLOGY, JAMAICA

Faculty Of Engineering and Computing

School Of Computing and Information Technology

CMP2019: Software Engineering: Analysis and Design

Tutor: Monique Daubon

Academic Year: 2023/2024

Semester: 2

Due Date: February 9, 2024

In five groups of three (3), select one of the following topics to research on using peer review articles.

Chosen topic:

3. Software Security Measures: With the increasing complexity of software systems, security remains a critical concern. Look at how the following can aid in software engineering:

- secure coding practice
- vulnerability detection
- threat modelling.

Definitions:

- Software Engineering – is the technological and managerial discipline concerned with the systematic production and maintenance of software products that are developed and modified on time and within budget.
- Software Security Measures - refers to the practice of protecting software applications, systems, and data from unauthorized access, vulnerabilities, and malicious activities. It encompasses a set of measures, processes, and best practices designed to ensure software and data confidentiality, integrity, and availability.
- Secure Coding - is the practice of developing computer software in such a way that guards against the accidental introduction of security vulnerabilities. This is done because defects, bugs, and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities.
- Vulnerability Detection – flaws in an application or computer system that causes it to act outside of its intended purpose.
- Threat Modelling - a systematic technique of recognizing and assessing potential threats in software or computer systems.
- **The OWASP Foundation -** works to improve the security of software through its community-led open-source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

How can 'Secure Coding Practices' aid in Software Engineering?

1. **Preventing Attacks by Malicious Users:**

   - According to Zenah and Abd Aziz (2011) Secure coding, as advocated by OWASP, provides examples and guidelines to prevent web applications from being attacked by

malicious users. It helps in identifying and addressing vulnerabilities in the early stages of development.

2. **Guideline for Web Application Development:**

   - Secure coding serves as a guideline for developers in building secure web applications. It is especially crucial in the absence of proper guidelines (Zenah & Abd Aziz, 2011), as it provides best practices and recommendations for creating robust and secure software.

3. **Reducing Risk at the Deployment Stage:**

   - Secure coding practices, when followed from the early stages of development, aim to reduce the risk of security vulnerabilities at the deployment stage. This contrasts with waiting for vulnerability scanning tools to identify and analyze vulnerabilities after deployment (Zenah & Abd Aziz, 2011).

4. **Cost-Effectiveness of Implementing Security Measures:** Emphasizing a preventive approach, the research cites the cost-effectiveness of implementing security measures compared to the expenses incurred in mitigating attacks, (Roshaidie et al., 2020). This cost analysis encourages a proactive stance toward software security.

5. **All in all:**

   - Secure coding practices integrate security considerations early in the development process, address common vulnerabilities, and employ a variety of strategies to enhance the security posture of software systems. These practices encompass both proactive measures and reactive responses to potential security threats (Roshaidie et al., 2020).

Conclusion:

In conclusion, the research articulates a comprehensive perspective on secure coding practices and their transformative impact on the discipline of computer science. Secure coding emerges not merely as a technical imperative but as a cultural shift, requiring education, vigilance, and a commitment to integrating security seamlessly into the fabric of

software development. As the digital landscape continues to evolve, the adoption of secure coding practices stands as a linchpin in fortifying software against the ever-present specter of cyber threats. This discussion serves as a testament to the imperative nature of secure coding practices in shaping a resilient and secure future for software engineering.

How can 'Vulnerability Detection' aid in Software Engineering?

What is Vulnerability Detection in Software Engineering?

Vulnerability detection in the context of software security refers to identifying flaws or weaknesses in computer systems - at the application layer, network layer or system layer - that could potentially be exploited by attackers. These vulnerabilities may cause the software to behave (and be used) in unintended ways contrary to its documented design, leading to security risks like, unauthorized access, manipulation, or theft of sensitive data.

Three ways Vulnerability Detection can aid in software security.

1. **Using Machine Learning Algorithms to detect vulnerable code units:**
    - A study conducted by Medeiros et al, 2020 considered applications of varying security concern levels (from highly critical to non-critical) via experimentation with the employment of machine learning algorithms. The aim was to analyse the effectiveness of the algorithm in identifying specific areas of code susceptible to security breaches. With the use of such technology software engineers can tailor development and prioritize efforts based on the criticality of the vulnerabilities detected. This approach allows for more targeted and efficient allocation of resources towards the development of more robust software security measures.

2. **Using specialized tools and methods for efficient vulnerability detection activities:**
    - It is important for software engineers to learn about and assess the uses, strengths, and weaknesses; differences and similarities of various vulnerability tools and methods that can be applied in detection tasks. This knowledge allows them to make adequately informed decisions when it comes to assessing and fixing software vulnerabilities, this process ultimately contributes to improving the effectiveness and efficiency of vulnerability detection efforts in software engineering (Amankwah et al, 2017).
3. **Employing static analysis techniques to reduce cost of vulnerability rectification:**
    - Static analysis is a defensive, preventive technique that detects vulnerabilities in web applications by assessing the source code before its deployment (or without executing the source code), the technique helps to mitigate potential vulnerabilities, thereby proactively reducing the risk of security breaches. By identifying weaknesses in the code early in the development process, this approach not only helps to enhance the security of software systems but also reduces the cost and effort required to rectify vulnerabilities later in the development lifecycle (Amankwah et al, 2017).

Vulnerability Detection tools

- Network scanners.
- Web application scanners
- Code scanners.
- Container scanners etc.

These tools perform automated or manual testing on various selected targets.

How can ‘Threat Modelling’ aid in Software Engineering?

What is Threat Modelling?

Threat modelling is regarded as one of the most critical operations in software security. Its goal is to identify all potential hazards and prevent or mitigate the consequences of threats and assaults on software systems. A systematic technique of recognizing and assessing threats (i.e. prospective assaults), which includes understanding threat agents' aims and adversaries' activities in attacking a system, based on that system's assets. (Messe et al, 2020). The primary concept underlying threat modelling is to think like an adversary throughout the design phase. A well-defined threat model assists in identifying threats to various assets of a system by utilizing well-grounded assumptions on the capabilities of any attacker interested in attacking such a system, thereby enabling software development teams to identify critical areas of design that need to be protected (Bernsmed et al,).

Threat modelling can occur at any point in the software development lifecycle, but it is most effective during the early requirement and architecture/design phases. The threat modelling method is also a collaborative process in which participants include business stakeholders, product team members from various product development phases, such as enterprise, software, and application architects, development leads, and so on. (Messe et al, 2020).

Ways threat modelling help in Software engineering

- Helps in identifying business-logic flaws and other critical vulnerabilities that expose core business assets
- Enriching assessments with new potential attack vectors
- Prioritizing type of attack to address

- Mitigating the risks more effectively
- Fixing issues early in development process

References

About the Owasp Foundation. About the OWASP Foundation | OWASP Foundation. (n.d.). https://owasp.org/about/

Amankwah, R., Kudjo, P. K., & Antwi, S. Y. (2017). Evaluation of software vulnerability detection methods and tools: a review. *International Journal of Computer Applications*, 169(8), 22-27. https://www.researchgate.net/profile/Richard-Amankwah/publication/318484922_Evaluation_of_Software_Vulnerability_Detection_Methods_and_Tools_A_Review/links/5a68107c4585159da0da0258/Evaluation-of-Software-Vulnerability-Detection-Methods-and-Tools-A-Review.pdf

Bernsmeda ,K, Cruzesa,D,S, Jaatuna,M,G & Iovan,M, M .Adopting threat modelling in agile software development projects (pp 1). https://jaatun.no/papers/2021/threat-modeling-journal-preprint.pdf.

Medeiros, N., Ivaki, N., Costa, P., & Vieira, M. (2020). Vulnerable code detection using software metrics and machine learning. *IEEE Access*, 8, 219174-219198. https://ieeexplore.ieee.org/abstract/document/9272730/

Messe, N, Chriproanov, V. Belloir,N. El-Hachem,J. Fleurquin,R. Sadou,S. (2020). Asset-Oriented Threat Modeling. https://hal.science/hal-02990919/document.

Roshaidie, M. D., Liang, W. P. H., Jun, C. G. K., & Yew, K. H. (2020). Importance of Secure Software Development Processes and Tools for Developers. *arXiv preprint arXiv:2012.15153*.

Zenah, N. H. Z., & Abd Aziz, N. (2011, December). Secure coding in software development. *Malaysian Conference in Software Engineering* (pp. 458-464). IEEE.