Security Testing

By Aarithi Rajendren

Problem Formulation

Chapter 9.3 of the Sommerville textbook discusses Security Testing, which aims to identify vulnerabilities (security risks) in a system and ensure it can resist attacks that could compromise its availability, integrity, or user data. Security risks can occur at any level during the software development life cycle (SDLC). An example of a security risk at each stage can be seen in Figure A.

Security Risk	Level in Software Development Cycle
Lack of developing threat modeling during the design phase	Design Phase
Tampering: Unauthorized modification of data	Coding Phase
Lack of Penetration Analysis Security Testing	Testing Phase
Lack of default software configuration	Deployment Phase
Lack of security trust	Maintenance Phase

Figure A: Security Risks in each Stage

Literature Search

- With 97.5% of security risks found during the Design phase of the SDLC, early security testing and validation is crucial to avoid the high costs and complexity of addressing issues after development (1, 2, 3)
- Threat modeling, static analysis, penetration testing, and vulnerability scanning are critical security testing practices that should be incorporated throughout the SDLC to build secure software (1)
- Resources like MITRE's CAPEC and OWASP's Top 10 serve as foundational references for threat modeling and testing, enabling development teams to anticipate, simulate, and defend against the most common and dangerous attack methods seen in real-world systems (2, 3).

Critical Analysis

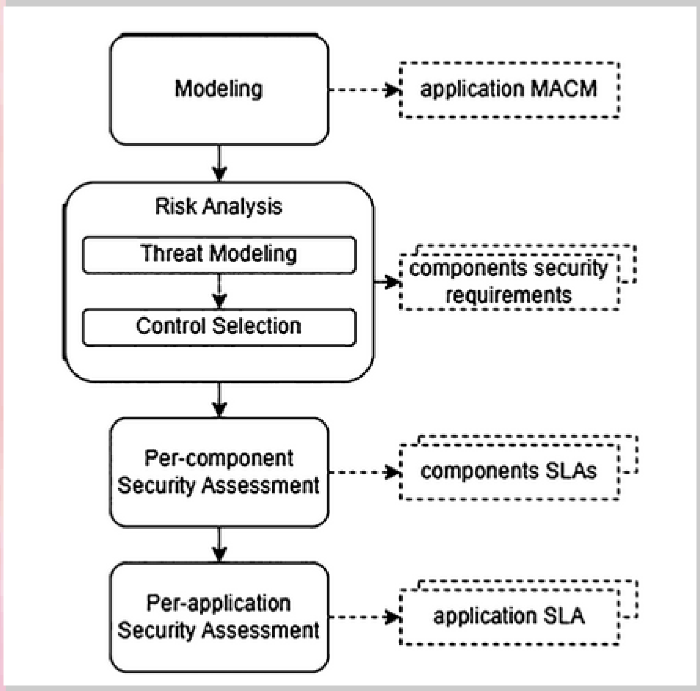
Modeling
application MACM
Risk Analysis
Threat Modeling
Control Selection
components security requirements
Per-component Security Assessment
components SLAs
Per-application Security Assessment
application SLA

Figure B: The SSDE Methodology

- The high reliance on automated tools (SAST, DAST) for security testing increases efficiency but introduces challenges, including a high rate of false positives. Developers might spend excessive time addressing alerts that don't represent genuine threats, diverting resources from actual vulnerabilities (3).
- Security testing, when applied early in the SDLC, significantly reduces both security vulnerabilities and overall development costs. However, the extensive upfront analysis required can initially slow down development, leading to potential productivity concerns (1)
- Well-organized security testing works much better than just guessing or using random tools. Figure B shows a step-by-step process that starts with planning and identifying risks, then checks each part of the software before reviewing the entire system — helping make sure everything is protected from the ground up (3).

Interesting Facts

- Most cyberattacks happen simply because systems were left unprotected — hackers just walk right in through easy-to-miss mistakes like weak passwords or unguarded entry points (3).
- Nearly 80% of successful system break-ins happen because of simple coding errors — things like forgetting to lock doors (figuratively) in the code (2).
- The OWASP Top 10 is like a hacker's checklist of the most common security flaws developers accidentally leave behind (1).

Evaluation

- The overwhelming presence of vulnerabilities in early stages like requirements and design reinforces the idea that security is not just a technical task, but a mindset that must be adopted from the beginning of a project.
- Across the literature, a recurring challenge is the gap between theory and practice—many smaller organizations lack the resources to implement recommended frameworks, which raises the need for more accessible, scalable security solutions.
- While automated tools are helpful for identifying vulnerabilities, their limitations—particularly false positives—highlight the continued need for trained professionals to interpret results and refine security strategies.

References

[1] R. A. Khan, S. U. Khan, H. U. Khan and M. Ilyas, "Systematic Literature Review on Security Risks and its Practices in Secure Software Development," in IEEE Access, vol. 10, pp. 5456-5481, 2022, doi: 10.1109/ACCESS.2022.3140181.
[2] Borky JM, Bradley TH. Protecting Information with Cybersecurity. Effective Model-Based Systems Engineering. 2018 Sep 9:345–404. doi: 10.1007/978-3-319-95669-5_10. PMCID: PMC7122347.
[3] Valentina Casola, Alessandra De Benedictis, Carlo Mazzocca, Vittorio Orbinato,
Secure software development and testing: A model-based methodology, Computers & Security, Volume 137, 2024, 103639, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2023.103639.