BSIMM Software Security Maturity Assessment

Application Security Case Study

A software security maturity assessment using BSIMM to evaluate governance, intelligence, secure development lifecycle touchpoints, and deployment practices, with prioritized recommendations for improvement.

BSIMM | Application Security | Secure SDLC | Threat Modeling | Code Review | SAST / DAST | Penetration Testing

AT A GLANCE

Scenario	A fictional university (course scenario) that recently suffered an application security incident and needs to understand its software security posture
Method	Map current development practices to the four BSIMM domains and their practices: current state, BSIMM alignment, recommendation
Finding	Early-stage maturity: compliance and development processes exist, but application security is informal and reactive
Outcome	A phased path focused first on governance, training and secure SDLC integration

Sean Richard

Portfolio edition based on original Full Sail University coursework. The assessed university is a fictional course scenario, not a real institution. Substantive project

sean@autom8ionlab.com · linkedin.com/in/a8l-sean-richard · +1.352.818.9694

content is preserved; classroom-only submission headers were removed.

INTRODUCTION

Why BSIMM, and what it measures

Following a recent security incident, the University in this case scenario has recognized the need to improve its software security posture. To better understand the current state of application security and identify areas for improvement, this assessment applies the Building Security In Maturity Model (BSIMM). BSIMM is a descriptive maturity model based on real-world observations from hundreds of organizations and is used to measure and guide software security initiatives.

This assessment evaluates the University's software development practices across the BSIMM domains and highlights gaps in governance, intelligence, secure development lifecycle touchpoints, and deployment. Recommendations are provided to help the University mature its software security program and reduce future risk.

Domain 1

Governance

Strategy & Metrics · Compliance & Policy · Training

Domain 2

Intelligence

Attack Models · Security Features & Design

Domain 3

SSDL Touchpoints

Architecture Analysis · Code Review · Security Testing

Domain 4

Deployment

Penetration Testing · Configuration & Vulnerability Management

Each domain contains practices and observable activities that indicate the maturity of an organization's software security initiative. By mapping the University's current practices to BSIMM activities, we can determine where improvements are most needed.

DOMAIN 1

Governance

Strategy and Metrics

CURRENT STATE The University currently lacks a formal application security strategy. Development teams prioritize feature delivery and deadlines over secure coding practices. While there may be individual developers interested in security, there is no executive sponsorship, formal metrics, or defined security objectives tied to the software development process.

BSIMM ALIGNMENT This reflects limited maturity in Strategy and Metrics, particularly activities related to defining security objectives, tracking progress, and integrating security into business decision-making.

RECOMMENDATION the University should establish a formal software security strategy supported by leadership. This includes defining measurable security goals such as vulnerability reduction, secure coding adoption, and incident response metrics. A designated security champion or small application security group should be empowered to guide these efforts and report progress to leadership.

Compliance and Policy

CURRENT STATE The university complies with regulatory requirements such as FERPA and GLBA; however, these controls focus more on data handling and infrastructure than application-level security. There is little evidence that compliance requirements are translated into secure coding standards or development policies.

BSIMM ALIGNMENT This maps to early-stage Compliance and Policy maturity, where policies exist but are not consistently enforced throughout the SDLC.

RECOMMENDATION the University should formalize secure development policies and ensure compliance requirements are explicitly mapped to application security controls. Policies should be documented, communicated to developers, and enforced through reviews and tooling.

Training

CURRENT STATE Developers do not receive structured secure coding training. Security knowledge is informal and inconsistent across teams, increasing the likelihood of recurring vulnerabilities.

BSIMM ALIGNMENT Training activities are largely absent, indicating a low maturity level in this practice.

RECOMMENDATION the University should introduce role-based security training for developers, testers, and architects. Training should cover common vulnerabilities, secure coding practices, and threat awareness, and be reinforced regularly through refreshers or hands-on exercises.

DOMAIN 2

Intelligence

Attack Models

CURRENT STATE There is no evidence of structured threat modeling, attack surface analysis, or attacker persona development. Security considerations are often reactive rather than proactive.

BSIMM ALIGNMENT Attack modeling activities are minimal, leaving applications vulnerable to predictable attack patterns.

RECOMMENDATION Threat modeling should be introduced early in the design phase of applications. Using frameworks such as STRIDE can help teams identify threats, understand attack paths, and design mitigations before development begins.

Security Features and Design

CURRENT STATE Security controls such as authentication, authorization, and input validation are implemented inconsistently. Design decisions often do not account for security risks until late in development or after an incident occurs.

BSIMM ALIGNMENT This reflects low maturity in secure design practices.

RECOMMENDATION the University should define standard security design patterns and reusable components for common functions like authentication and access control. Security reviews should be required during the architecture and design phases of new projects.

DOMAIN 3

Secure Software Development Lifecycle Touchpoints

Architecture Analysis

CURRENT STATE Architectural reviews focus on functionality and performance, with little emphasis on security risk. Security is not consistently considered during system design.

BSIMM ALIGNMENT Architecture analysis activities are limited or informal.

RECOMMENDATION Security architecture reviews should be integrated into the SDLC. These reviews should assess trust boundaries, data flows, and potential attack vectors before implementation.

Code Review

CURRENT STATE Code reviews occur, but they primarily focus on correctness and style rather than security. There is no standardized checklist or tooling for identifying security flaws.

BSIMM ALIGNMENT This indicates early-stage maturity in secure code review.

RECOMMENDATION Secure code review checklists should be introduced, and developers should be trained to identify common vulnerabilities. Automated static analysis tools can further enhance code review effectiveness.

Security Testing

CURRENT STATE Testing is focused on functionality, with minimal security testing performed. Vulnerabilities are often discovered after deployment.

BSIMM ALIGNMENT Security testing maturity is low.

RECOMMENDATION the University should incorporate security testing into the QA process. This includes static analysis, dynamic testing, and vulnerability scanning prior to release. Security defects should be tracked and prioritized alongside functional bugs.

DOMAIN 4

Deployment

Penetration Testing

CURRENT STATE Penetration testing is not performed regularly and is typically reactive after incidents.

BSIMM ALIGNMENT This reflects limited maturity in penetration testing practices.

RECOMMENDATION Regular penetration testing should be conducted on high-risk applications. Findings should feed back into development and training efforts to prevent repeat issues.

Configuration Management and Vulnerability Management

CURRENT STATE There is limited visibility into application vulnerabilities post-deployment, and patching may be inconsistent.

BSIMM ALIGNMENT Vulnerability management activities are underdeveloped.

RECOMMENDATION the University should implement centralized vulnerability tracking and establish clear processes for remediation. Monitoring tools and patch management procedures should be standardized across environments.

CONCLUSION

Applying the BSIMM framework reveals that the University is in the early stages of software security maturity. While compliance and development processes exist, application security is largely informal and reactive. By adopting BSIMM-aligned practices such as formal governance, secure design, threat modeling, training, and security testing, the University can significantly reduce risk and improve resilience against future incidents. A phased approach focusing first on governance, training, and secure SDLC integration will provide the strongest foundation for long-term improvement.

References

Building Security In Maturity Model (BSIMM). (2022). *BSIMM13 foundations.* Synopsys. https://www.bsimm.com

National Institute of Standards and Technology. (2022). *Secure software development framework (SSDF)*(NIST SP 800-218). https://www.nist.gov/itl/ssd/software-quality-group/secure-software-development-framework

Open Web Application Security Project. (2021). *OWASP top 10 web application security risks.* https://owasp.org/Top10/

Veracode. (2023). *State of software security 2023.* https://www.veracode.com/state-of-software-security