Toward Building and Validating a Secure Software Development Self-Efficacy Scale

Desiree Abrokwa, Daniel Votipka, and Michelle L. Mazurek

Goal

- Build and validate a Secure Software Development Self-Efficacy survey to measure a developer's confidence in completing secure development tasks

Motivation

- Prior security education measures can be time consuming/noisy and are difficult to measure their effectiveness

Item Generation

Frameworks Used

- NIST's NICE Framework
- Building Security in Maturity Model
- Open Web Application Security Project
- Software Assurance Forum for Excellence in Code
- Fundamental Practices for Secure Software Development

Expert Review

- Surveyed 22 security professionals with an average of 20.5 years of secure development experience on the applicability of our scale and tasks

Survey

- Each of the 58 tasks created fit into one of these categories

Determining Security Requirements

- Identify potential security threats to the system
- Determine security requirements for the system

Identifying Attack Vectors

- Identify potential attack vectors associated with the system under development
- Identify potential attack vectors in environment the system interacts with (e.g., hardware, libraries, etc.)

Identifying Vulnerabilities

- Identify potential vulnerabilities as you write code
- Identify common vulnerabilities of a programming language

Implementing Mitigations

- Use secure implementations of common libraries
- Identify secure implementations of common libraries

Testing

- Enumerate boundary conditions and mimic potential threats
- Assess that security requirements are met (e.g., through security design and code reviews)

Communicating Security

- Communicate system details with other developers to ensure a thorough security review of the code
- Maintain awareness of security issues with new hardware and software technologies and their potential implications

- Using a 5-item Likert Scale *from "I am not confident at all"* to *"I am absolutely confident"*, 157 developers were surveyed on how well they can complete these tasks
- Participants recruited from various platforms

Survey Results

	Tasks	Mean	Standard Deviation
Security- Specific Questions	I can perform a threat risk analysis (e.g., likelihood of vulnerability and impact of exploitation).	3.17	1.27
	I can identify potential security threats to the system.	3.61	1.09
	I can identify the common attack techniques used by attackers.	3.44	1.13
	I can identify potential attack vectors in the environment the system interacts with (e.g., hardware and libraries).	2.98	1.23
	I can identify common vulnerabilities of a programming language.	3.45	1.13
	I can design software to quarantine an attacker if a vulnerability is exploited.	2.69	1.34
	I can mimic potential threats to the system.	3.21	1.18
	I can evaluate security controls on the system's interfaces/interactions with other software systems.	3.29	1.22
	I can evaluate security controls on the system's interfaces/interactions with hardware systems.	2.93	1.28
Non-Security Specific Questions	I can identify code that handles sensitive data (e.g., Personally Identifiable Information).	4.08	1.09
	I can correctly implement authentication protocols.	3.76	1.15
	I can correctly implement authorization protocols.	3.76	1.12
	I can communicate security assumptions and requirements to other developers on the team to ensure vulnerabilities are not introduced due to misunderstandings.	3.70	1.08
	I can communicate system details with other developers to ensure a thorough security code review.	3.82	1.17
	I can discuss lessons learned from internal and external security incidents to ensure all development team members are aware of potential threats.	3.87	1.11
	I can effectively communicate to company leadership identified security issues and the cost/risk trade-off associated with deciding whether or not to fix the problem.	3.78	1.13
	I can communicate functionality needs to security experts to get recommendations for secure solutions (e.g., secure libraries, languages, design patterns, and platforms).	3.81	1.10
	I know the appropriate point of contact/response team in my organization to contact if a vulnerability in production code is identified.	4.01	1.24

- Removed poor performing questions (e.g., floor/ceiling effects, poor item-total correlation)
- Used explanatory factor analysis to identify underlying factor structure

Future Work

- Recruit 120 more subjects to complete the survey and validate the underlying scale factor structure
- Compare to other psychometric measures
- Use as a before and after metric for future studies of intervention effectiveness

University of Maryland, Maryland Cybersecurity Center sec-professionals.cs.umd.edu