SCIPP INTERNATIONAL™
THE SECURITY AWARENESS CERTIFICATION COMPANY

1964 Gallows Road, Suite 320
Vienna, Virginia 22182
United States of America
+1 703.637.4433
sales@scippinternational.org
www.SCIPPinternational.org

SECURE WEB APPLICATION DEVELOPMENT AWARENESS (SWADA™)

DEVELOPER PRINCIPLES

SCIPP SWADA - Developer Principles Generally Accepted Practices (GAP™)	Competencies
1.0 Security Aspects of the Software Development Life Cycle (SDLC)	**Objective: To raise awareness of the SDLC processes that promote software security.**
1.1 Security Originates Within the SDLC	Articulate how a properly controlled SDLC facilitates the secure development of software.
1.2 SDLC Phase 0 - Developer Training	Describe how the life cycle begins with fundamental training of the developer in secure software development.
1.3 Requirements Gathering and Analysis	Identify the security related issues pertinent to the initial phase of gathering and analyzing the software requirements.
1.4 Systems Design	Understand the security facets during system design of the software system.
1.5 Design Reviews	Describe the function that design review fulfills in the SDLC.
1.6 Static Analysis	Use automated tools to find several types of issues within source code.
1.7 Development Phase	Articulate the important security considerations relating to the development phase of the SDLC, especially with respect to peer review.
1.8 Testing Phase	Identify the security components germane to the testing of code and applications.
1.9 Deployment Phase	Describe the security related aspects when deploying new software.
1.10 Learning Points	Summarize the phases of the software development life cycle and subsequent activities that set the stage for establishing and defining security and software quality requirements.
2.0 Proven Best Practices For Secure Software Development	**Objective: To raise the awareness of best practices for defensive programming.**
2.1 Best Practices Concept	Articulate the concepts that underpin the need for software development best practices.
2.2 Attack Surface	Describe how the code within a computer system that can be run by unauthorized users is defined.
2.3 Security Perimeter	Demonstrate an understanding of how physical and programming security policies provide levels of protection against remote malicious activity.
2.4 Best Practices for Secure Development	Summarize the important Best Practices during the development of new software.
2.5 Learning Points	Summarize the general best practices that should be followed during software development.
3.0 Design Phase Security Activities	**Objective: To raise the awareness of key security principles which apply during the design phase of software development.**
3.1 Industry Recommendations for Secure Software Designs	Describe the important information security industry recommendations for secure software designs.
3.2 Meeting Design Security Requirements	Articulate how security is built into the functional requirements of the design to ensure that software behaves as it should, but does it securely.
3.3 Design Patterns	Describe how general reusable solutions commonly occurring problems can be employed within a given context in software design.
3.4 Architecting Web Applications via Design Patterns	Summarize the important special security considerations when architecting Web applications. Use SEI report as a model.
3.5 Learning Points	Summarize the important security principles that should be applied during the designing of a new application.

4.0 Vulnerability Taxonomies	**Objective: To raise the awareness of industry software vulnerabilities, threats, and attack taxonomies.**
4.1 Overview of Available Classification Systems	Identify the available methods used to classify threats, vulnerabilities, and exploits.
4.2 Learning Points	Summarize the differences between vulnerabilities, threats and exploits, the current ways in which they are named, classified and tracked, and the methods for developing and testing secure code.
5.0 OWASP Top 10	**Objective: To raise awareness of the most frequent types of software vulnerabilities.**
5.1 Compare 2010 Version with 2013	The report is reissued every 3 years, with minimal changes. Compare the differences.
5.2 Learning Points	Summarize the ten most common types of software vulnerabilities.
6.0 CWE/SANS Top 25 Most Dangerous Programming Errors	**Objective: To raise the awareness about the kinds of programming errors which are the most dangerous.**
6.1 Guidance For Use By Audiences:	Describe the list of multiple audiences for which the list is intended and the varying security skill levels of the audiences.
6.2 Category: Insecure interactions between Components	Demonstrate an understanding of the 6 types of common programming errors that involve insecure interactions between software components.
6.3 Category: Risky Resource Management	Demonstrate an understanding of the 8 types of common programming errors that involve risky resource management.
6.4 Category: Porous Defenses	Demonstrate an understanding of the 11 types of common programming errors that involve porous defenses.
6.5 Learning Points	Summarize and give examples of each of the categories of programming errors: resources management, defenses and interaction between components.
7.0 WASC Threat Classification V2	**Objective: To raise awareness of how threats are categorized.**
7.1 WASC Threats – Data Views	Explain the types of Data Views and their corresponding purposes.
7.2 Learning Points	Summarize the threat categories, and give examples of how the assignment of threats to categories is determined.
8.0 Programming Best Practices	**Objective: To raise the awareness concerning the most important Best Practices for programming software.**
8.1 Input Validation	List and describe the top ten best practices for input validation.
8.2 Secure Coding	List and describe the top ten best practices for secure coding.
8.3 Monster Mitigations	List and describe the top ten mitigations that have the significant impact, as described by SANS under the name “Monster Mitigations”.
8.4 Learning Points	Summarize leading best practices for secure programming which lead to secure applications written with the highest quality.
9.0 Testing	**Objective: To raise the awareness of the appropriate methods for testing, validating and verifying the security of an application.**
9.1 Three Kinds of Security Testing	Describe the types of tools and methods that are employed to scan source code first, and then object code.
9.2 Testing Your Code	Explain the benefits and weaknesses involved when coders test their own code.
9.3 Use and Misuse Testing	Explain how these two forms of testing work, their differences, and the benefits to be derived from each.
9.4 OWASP Web Scarab	Introduce the framework from OWASP dedicated to uncovering web (HTTP/HTTPS) weaknesses.
9.5 Survey Vendors	Briefly describe some of the vendors of Source Code Analysis and Penetration Testing tools.
9.6 Penetration Testing	Describe the types of penetration tests and the types of testers.
9.7 Training	Introduce OWASP’s training tool, Web Goat, and how it can be used to help developers avoid the Top 10 pitfalls.
9.8 Learning Points	Summarize the types of appropriate security testing methods and the benefits for testing, validating and verifying the security of an application.
10.0 Enterprise Security API (ESAPI)	**Objective: To raise the awareness about how the Open Web Application Security Project (OWASP) ESAPI project contributes to enterprise security.**
10.1 Overview of ESAPI	Describe the ESAPI architecture.
10.2 ESAPI Benefits	Show a sample before and after model.
10.3 Map OWASP Top 10 to ESAPI	Show how the ESAPI architecture addresses the Top 10 Vulnerabilities.
10.4 Learning Points	Summarize the Enterprise Security API (ESAPI) project, and how it is used to ensure enterprise-level security.