Software Assurance (SwA) in Education, Training & Certification

Pocket Guide v2.1

Robin A. Gandhi
Nebraska University Center on Information Assurance (NUCIA)
University of Nebraska at Omaha
What is a Pocketguide?

• Self-contained
• Concise
• Enumeration of resources
• Theme
• Living document
• Reprints and redistribution possible
• Fits in the coat pocket
SwA ETC Pocketguide Theme

• Educating the Educator/Trainer on available SwA resources

• Purpose:
  – Awareness resource for “getting started” in educating, training and sustaining a workforce capable of producing secure software
  – An “index” in to a vast amount of resources, tools, curricula, and certification and training opportunities for software assurance
Software Assurance Pocket Guide Resources

This is a resource for getting started in educating, training and certifying a workforce to build secure software. It describes how to promote awareness of the engineering activities and knowledge areas needed to build software that operates as expected, free from vulnerabilities. It summarizes how to train to prevent vulnerabilities from being intentionally designed into the software or accidentally inserted at any time during its life cycle. To do so, this guide describes knowledge areas for software assurance, starting with the core areas of study and extending to sub-disciplines to enhance with software security subject materials. It then presents lists of resources for accomplishing such study, including programs, tools, and books, with pointers on their use. Lastly, this guide describes the people who make up a security-conscious system development team, their education, titles, credentials, and standards. As part of the Software Assurance (SwA) Pocket Guide series, this resource is for information only. For details, see referenced source documents. For proper attribution, please include mention of these sources when referencing any part of this document.

This volume of the SwA Pocket Guide series focuses on enumerating education, training, and certification resources. It identifies the most effective strategies to inject software assurance topics into existing college curriculums and workforce training and certification programs.

At the back of this pocket guide are reprints, limitation statements, and a listing of topics addressed in the SwA Pocket Guide series. All SwA Pocket guides and SwA-related documents are freely available for download via the SwA Community Resources and Information Clearinghouse at https://cise.coe.ou.edu/swa.

Acknowledgements

The SwA community collaborates to develop SwA Pocket Guides. The SwA Forum and Working Groups function as a stakeholder meta-community that welcomes additional participation in advancing and refining software security. All SwA-related information resources are offered free for public use. The SwA community invites your input: please contact Software Assurance@cis.ou.edu for comments and inquiries. For the most current pocket guides, refer to the SwA website at https://cise.coe.ou.edu/swa.

Members from government, industry, and academia comprise the SwA Forum and Working Groups. The Groups focus on incorporating SwA considerations into acquisition and development processes to manage potential risk exposure from software and from the supply chain.

Participants in the SwA Forum’s Workforce Education and Training Working Group contributed to developing the material used in this pocket guide as a step in raising awareness on how to incorporate SwA topics in education, training, and certification of a knowledgeable workforce. One that is ready to perform engineering and technical activities that promote software assurance throughout the Software Development Life Cycle (SDLC).
Software Assurance?

• The *basis for the belief* that software will work as expected
  – Claims, arguments, evidences that span the software lifecycle from cradle to grave
  – People, Process, Technology that enable us to promote assurances in the software that is mission and business critical
Table of Contents

OVERVIEW .................................................................................................................. 3
THE CASE FOR SOFTWARE ASSURANCE EDUCATION .................................................. 5
KEY SWA KNOWLEDGE AREAS AND EFFORTS .......................................................... 6
CURRICULUM AND TRAINING GUIDES .................................................................... 7
WORKFORCE DEVELOPMENT AND IMPROVEMENT ............................................... 8
STRATEGIES FOR INJECTING SWA KNOWLEDGE ................................................. 8
TOOLS ....................................................................................................................... 10
BOOKS .................................................................................................................. 12
STANDARDS OF PRACTICE .................................................................................. 13
WORKFORCE CREDENTIALS .................................................................................. 14
VENDORS ............................................................................................................... 15
ROLE DESCRIPTIONS ............................................................................................ 17
CONCLUSION ......................................................................................................... 20
REPRINTS ............................................................................................................... 21
SwA Knowledge Areas and Efforts

Key Software Assurance Knowledge Areas and Efforts

- **Workforce Education and Training**
  - Curriculum Guides
  - Knowledge necessary to Develop Sustain, Acquire and Assure Secure Software (SwABoK)
  - State of the Art Reports (SOAR)
  - Guidebooks (NASA, DACS)
  - Reference Curriculum (MSwA2010, Undergrad outline)
  - Security Principles and Guidelines
  - Logical and in-depth organization of Principles and Guidelines
  - Workforce Development and Improvement
  - Competency and Functional Framework for IT Security Workforce (EBK)
  - Workforce Credentials

- **Technology, Tool and Product Evaluation**
  - Making Security Measureable
  - CWE, CVSS
  - CVE, CCE, CPE, OVAL, CVSS
  - Measuring Functionality and Capability of SwA Tools (SAMATE)
  - Functional Specifications
  - Test suites
  - Tool Metrics
  - Metamodels for Software Assets and Operational Environments
  - Abstract Syntax Tree Metamodel (ASTM)
  - Knowledge Discovery Metamodel (KDM)
  - Software Metrics Metamodel (SMM)

- **Acquisition and Outsourcing**
  - Security-Enhanced Software Acquisition and Outsourcing
    - Reference Guide: SwA in Acquisition
    - Practices to Enhance SwA in Purchasing
    - Due diligence Questionnaires
    - Sample Contract Provisions and Language
    - Application Security Procurement Language
    - Supply chain Risk Management
    - Risk-based approach to Software Acquisition

- **Malware**
  - Malware Dictionaries
  - Malware Attribute Enumeration and Characterization (MAEC)
  - Novel Approaches to Malware

- **Processes and Practices**
  - Enhancement of Development Lifecycle
  - Integrating Security into the Software Development Lifecycle
  - Requirements and Analysis
  - Architecture and Design Considerations
  - Risk-Based Security Testing
  - Capability Maturity Model Integration
  - Harmonizing and Extending existing Security Capability Maturity Models
  - Mapping Assurance Goals and Practices to CMMI for Development
  - Maturity Model
    - Building Security In Maturity Model (BSIMM)
    - Software Assurance Maturity Model (SAMM)
  - Key Practices for Mitigating Software Weaknesses
    - Secure Coding Standards (CERT)

- **Measurement**
  - Measurement Frameworks
    - Practical Measurement Framework for Software Assurance and Information Security
    - Acquisition Measurement
    - Measurements Goals and Questions Lists

- **Business Case**
  - Making a Business Case for SwA
    - Cost/Benefit Models Measurement
    - Risk
    - Prioritization
    - Process Improvement
    - Globalization
    - Organizational Development
    - Case Studies and Examples
SwA Knowledge Areas and Efforts

Workforce Education and Training
- Curriculum Guides
  - Knowledge necessary to Develop, Acquire and Assure Secure Software (SwABoK)
  - State of the Art Reports (SOAR)
  - Guidebooks (NASA, DACS)
  - Reference Curriculum (MSwA2010, Undergrad outline)
- Security Principles and Guidelines
  - Logical and In-depth organization of Principles and Guidelines
- Workforce Development and Improvement
  - Competency and Functional Framework for IT Security Workforce (EBK)
  - Workforce Credentials

Technology, Tool and Product Evaluation
- Making Security Measureable
  - CVE, CWSS
- Measuring Functionality and Capability of SwA Tools (SAMATE)
- Functional Specifications
- Test suites
- Tool Metrics
- Metamodels for Software Assets and Operational Environments
  - Abstract Syntax Tree Metamodel (ASTM)
  - Knowledge Discovery Metamodel (KDM)
  - Software Metrics Metamodel (SMM)

Processes and Practices
- Enhancement of Development Lifecycle
  - Integrating Security into the Software Development Lifecycle
  - Requirements and Analysis
  - Architecture and Design Considerations
  - Risk-Based Security Testing
- Capability Maturity Model Integration
  - Harmonizing and Extending existing Security Capability Maturity Models
  - Mapping Assurance Goals and Practices to CMMI for Development
- Maturity Model
  - Building Security In Maturity Model (BSIMM)

Acquisition and Outsourcing
- Security-Enhanced Software Acquisition and Outsourcing
  - Reference Guide: SwA in Acquisition
  - Practices to Enhance SwA in Purchasing
  - Due diligence Questionnaires
  - Sample Contract Provisions and Language
  - Application Security Procurement Language
- Supply chain Risk Management
- Risk-based approach to Software Acquisition
The Various WGs and Deliverables

**Measurement Frameworks**
- Practical Measurement Framework for Software Assurance and Information Security
- Acquisition Measurement
- Measurements Goals and Questions Lists

**Business Case**
- Making a Business Case for SwA
- Cost/Benefit Models
- Measurement
- Risk
- Prioritization
- Process Improvement
- Globalization
- Organizational Development
- Case Studies and Examples

**Key Software Assurance Knowledge Areas and Efforts**

**Processes and Practices**
- Enhancement of Development Lifecycle
- Integrating Security into the Software Development Lifecycle
- Requirements and Analysis
- Architecture and Design Considerations
- Risk-Based Security Testing

- Capability Maturity Model Integration
- Harmonizing and Extending existing Security Capability
- Maturity Models
- Mapping Assurance Goals and Practices to CMMI for Development

**Key Practices for Mitigating Software Weaknesses**
- Secure Coding Standards (CERT)

**Security Principles and Guidelines**
- Logical and In-depth organization of Principles and Guidelines

**Workforce Development and Improvement**
- Competency and Functional Framework for IT Security Workforce (EBK)
- Workforce Credentials

**Metamodels for Software Assets and Operational Environments**
- Abstract Syntax Tree Metamodel (ASTM)
- Knowledge Discovery Metamodel (KDM)
- Software Metrics Metamodel (SMM)

**Malware**
- Malware Dictionaries
- Malware Attribute Enumeration and Characterization (MAEC)

**Novel Approaches to Malware**

**Risk**
- Prioritization
- Process Improvement
- Globalization
- Organizational Development
- Case Studies and Examples

**Malware Procurement**
- Due diligence Questionnaires
- Sample Contract Provisions and Language

**Supply chain Risk Management**

**Acquisition and Outsourcing**
- Security-Enhanced Software Acquisition and Outsourcing
- Reference Guide: SwA in Acquisition
- Practices to Enhance SwA in Purchasing
- Sample Contract Provisions and Language
- Application Security Procurement Language
- Supply chain Risk Management

**Key Software Assurance Knowledge Areas and Efforts**
- CVE, CCE, CPE, OVAL, CVSS
- Measuring Functionality and Capability of SwA Tools (SAMATE)
- Functional Specifications
- Test suites
- Tool Metrics
# Curriculum and Training Guides

## Table 1 – SwA Curriculum and Training Development Guides

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Relevant Documents and Links</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SwA Curriculum Project</strong></td>
<td><strong>Volume I: Master of Software Assurance Reference Curriculum.</strong> Mead, Nancy R. et al. SEI/CMU. <a href="http://www.cert.org/mswa/">http://www.cert.org/mswa/</a>; <a href="http://www.cert.org/podcast/show/20101026mead.html">http://www.cert.org/podcast/show/20101026mead.html</a></td>
<td>Offers a core body of knowledge from which to create a master’s level degree program in software assurance, as a standalone offering and as a track within existing software engineering and computer science master’s degree programs. Last updated 2010.</td>
</tr>
<tr>
<td><strong>SwA CBK and Organization of Principles and Guidelines</strong></td>
<td><strong>Software Assurance Body of Knowledge. Version 1.2.</strong> Samuel T. Redwine, Jr. (Editor), DHS, <a href="https://buildsecurityin.us-cert.gov/bsi/dhs/927-BSI.html">https://buildsecurityin.us-cert.gov/bsi/dhs/927-BSI.html</a></td>
<td>Provides a comprehensive set of principles and guidelines from the disciplines of software engineering, systems engineering, information system, computer science, safety, security, testing, information assurance, and project management. Last updated October 2007.</td>
</tr>
</tbody>
</table>
# Workforce Development and Improvement

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Relevant Documents and Links</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Security Workforce Development Matrix – Information Security Systems and Software Development Professional Role</td>
<td>See Section <strong>Role Descriptions</strong> section, page 17, for materials from the matrix project developed by Federal CIO Council’s IT Workforce Committee and Information Security and Identity Management Committee. Contacts: <a href="http://www.cio.gov/committees.cfm/csec/3/cid/4">http://www.cio.gov/committees.cfm/csec/3/cid/4</a> <a href="http://www.cio.gov/committees.cfm/csec/3/cid/5">http://www.cio.gov/committees.cfm/csec/3/cid/5</a></td>
<td>This project produces role-based information security workforce development matrices. The matrices are one-page dashboards intended to establish a baseline across the Federal Government for staff engaged in information security work. This initiative provides a government-wide perspective on common information security roles. The ISS&amp;SDP is one of 11 roles that have been identified to date. Each matrix contains a uniform framework, by performance level, describing the recommended competencies/skills, education, experience, credentials, and training for a particular role. The matrices provide guidance for federal agencies and do not replace OPM basic qualifications.</td>
</tr>
</tbody>
</table>
## Strategies for Injecting SwA Knowledge

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Relevant Documents and Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree programs and specializations in SwA</td>
<td>Reference curriculums available from the Software Engineering Institute, Carnegie Mellon University can be used as recommendations for designing Masters of Software Assurance degree program and undergraduate curriculum specialization in software assurance. These reference curriculums are available at <a href="http://repository.cmu.edu/sei/3/">http://repository.cmu.edu/sei/3/</a> and <a href="http://repository.cmu.edu/sei/4/">http://repository.cmu.edu/sei/4/</a>. Graduate Certificates and Master Degree Concentrations at the Stevens Institute of Technology: <a href="http://dc.stevens.edu/academic-programs/software-assurance/">http://dc.stevens.edu/academic-programs/software-assurance/</a></td>
</tr>
<tr>
<td>Community Support</td>
<td><strong>LinkedIn SwA Education Discussion Group</strong>  &lt;br&gt;Nancy Mead, SwA Curriculum Team lead  &lt;br&gt;The objective of the SwA Curriculum Development Team in establishing this group is to provide a venue for dialog about software assurance education.  &lt;br&gt;<a href="http://www.linkedin.com/groups?mostPopular=&amp;gid=3430456">http://www.linkedin.com/groups?mostPopular=&amp;gid=3430456</a>  &lt;br&gt;<strong>Software Assurance Mobile Instructional (device) SAMI</strong>  &lt;br&gt;Dan Shoemaker, University of Detroit Mercy  &lt;br&gt;A catalogue of available software assurance materials that are packaged and delivered using an iPad based instruction medium for educators</td>
</tr>
<tr>
<td>Credentialing</td>
<td>Several certification options are now available to suit the needs of specific job functions required in an enterprise. More information can be found in the Workforce Credentials section of this guide.</td>
</tr>
<tr>
<td>Strategy</td>
<td>Relevant Documents and Links</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Stand-alone Courses            | New course offerings based on SwA knowledge areas complement existing Software Engineering courses. Examples: [http://www.cs.jmu.edu/sss](http://www.cs.jmu.edu/sss)  
[https://www.securecoding.cert.org/confluence/display/sci/S08+15392+Secure+Programming](https://www.securecoding.cert.org/confluence/display/sci/S08+15392+Secure+Programming)  
Also: graduate-level Software Assurance courses that cover secure software engineering activities during the SDLC are offered at the University of North Carolina at Charlotte, and The University of Nebraska at Omaha. |
| Augmenting Existing Courses    | The SwA CBK and State-of-the-Art reports are catalogs of secure software development practices, processes, and techniques that can be mapped to topics relevant to current curriculums. The identified gaps can then be filled using relevant materials. |
| Micro-Modules                  | Problem-based learning exercises, in class workshops, or short talks to inject topics such as Misuse Cases and Assurance Cases into existing software engineering or information security courses. |
| Capstone and Class Projects    | Software Engineering capstone courses or class projects which can be geared towards a security critical domain such as designing a software system for the Department of Defense, Cyber-physical systems or for a Credit Card transaction processing company. These domains will facilitate the exploration of security needs throughout the SDLC. |
| Online Courses                 | The CERT Virtual Training Environment (VTE) combines the components of traditional classroom training with the convenience of web-based training. Over 200 hours of course material focused around the technical, policy, and management implications of information security – including preparatory courses for commercial certifications, core skills courses, role-based courses for managers and technical staff, and vendor-developed courses. Open access is provided to individual DoD personnel (Active Duty, DoD Civilian and contractors) and members of the Federal Civilian Workforce through specific sponsorships from DISA, and DHS in conjunction with the Department of State Foreign Service Institute. Sponsored accounts can be requested at [www.vte.cert.org](http://www.vte.cert.org). Public access to many of the materials is provided through the VTE Library at [https://www.vte.cert.org/vteweb/Library/Library.aspx](https://www.vte.cert.org/vteweb/Library/Library.aspx) |
Tools and web resources that can be used in class to provide hands-on experience with SwA Concepts.

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Tool Description</th>
<th>Possible Classroom Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgoUML</td>
<td>ArgoUML is the leading open source UML modeling tool and includes support for all standard UML 1.4 diagrams. It runs on any Java platform.</td>
<td>Misuse cases, security focused UML class diagrams and other documentation for class assignments and projects.</td>
</tr>
<tr>
<td>Burp Suite</td>
<td>Burp Suite is an integrated platform for attacking web applications. Located at <a href="http://www.portswigger.net/suite/">http://www.portswigger.net/suite/</a></td>
<td>Burp Suite allows the combination of manual and automated techniques to enumerate, analyze, scan, attack and exploit web applications.</td>
</tr>
<tr>
<td>CERT Secure Coding Standards</td>
<td>Secure coding standards for commonly used programming languages such as C, C++ and Java. Located at <a href="https://www.securecoding.cert.org">https://www.securecoding.cert.org</a></td>
<td>Online reference; examples of coding do’s and don’ts.</td>
</tr>
<tr>
<td>FindBugs™</td>
<td>A program which uses static analysis to look for bugs in Java code at <a href="http://findbugs.sourceforge.net/">http://findbugs.sourceforge.net/</a></td>
<td>Scan java code repositories for bugs; Introduction to static code checking activities.</td>
</tr>
<tr>
<td>Microsoft SDL Threat Modeling Tool</td>
<td>The Microsoft SDL Threat Modeling Tool allows software architects to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve. Located at <a href="http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx">http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx</a></td>
<td>Conduct student group workshops to discuss threats to various design alternatives, while suggesting possible mitigation strategies.</td>
</tr>
<tr>
<td>Tool Name</td>
<td>Tool Description</td>
<td>Possible Classroom Uses</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Olly Debug</td>
<td>OllyDbg is a 32-bit assembly level debugger for Microsoft Windows. Located at <a href="http://www.ollydbg.de/">www.ollydbg.de/</a></td>
<td>Emphasize binary code analysis and is particularly useful in cases where source is unavailable. Explain Buffer Overflows.</td>
</tr>
<tr>
<td>Pharos</td>
<td>Pharos is an open source proxy that traps all HTTP and HTTPS data between server and client, including cookies and form fields, which can be intercepted and modified. Located at <a href="http://parosproxy.org/index.shtml">http://parosproxy.org/index.shtml</a></td>
<td>Pharos can be used as an introduction to web application security assessment.</td>
</tr>
<tr>
<td>SAMATE Reference Dataset</td>
<td>The purpose of the SAMATE Reference Dataset (SRD) is to provide users, researchers, and software security assurance tool developers with a set of known security flaws. This will allow end users to evaluate tools and tool developers to test their methods. Located at <a href="http://samate.nist.gov/index.php/Main_Page.html">http://samate.nist.gov/index.php/Main_Page.html</a></td>
<td>A reference data set can be used in class to reflect upon known flaws in software.</td>
</tr>
<tr>
<td>SDMetrics</td>
<td>Analyze the structural properties of UML models using object-oriented measures of design size, coupling, and complexity. Located at <a href="http://www.sdmetrics.com/">http://www.sdmetrics.com/</a></td>
<td>Examine object-oriented metrics and measures for design and source code artifacts.</td>
</tr>
<tr>
<td>Splint</td>
<td>Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes. Located at <a href="http://www.splint.org/">http://www.splint.org/</a></td>
<td>Static analysis code checking activities.</td>
</tr>
<tr>
<td>Valgrind</td>
<td>Valgrind is an instrumentation framework for building dynamic analysis tools. Located at <a href="http://valgrind.org/">http://valgrind.org/</a></td>
<td>Demonstrate dynamic analysis techniques to detect memory management and threading bugs, as well as detailed program profiling.</td>
</tr>
<tr>
<td>Vine</td>
<td>Provides an intermediate language that x86 code can be translated to for Static analysis. Located at <a href="http://bitblaze.cs.berkeley.edu/vine.html">http://bitblaze.cs.berkeley.edu/vine.html</a></td>
<td>Identify data flows analysis and conduct binary analysis.</td>
</tr>
</tbody>
</table>

Web Resources

- Google Code: http://google.grueve-appspot.com
- Web application exploits and defenses: http://www.rat-fish.net
<table>
<thead>
<tr>
<th>Topic</th>
<th>Title and Publisher</th>
<th>Summary and Possible Use</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Building Secure Software: How to Avoid Security Problems the Right Way</strong>, John Viega and Gary McGraw, Addison Wesley, 2002</td>
<td>Software Assurance principles and guidelines and Implementation level issues Possible use as a textbook or additional reference material</td>
</tr>
<tr>
<td>Standard</td>
<td>Source</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
Self Assessment: [https://buildsecurity.us-cert.gov/swa/downloads/20100922_PRM_Practice_List_2page.pdf](https://buildsecurity.us-cert.gov/swa/downloads/20100922_PRM_Practice_List_2page.pdf) | The Assurance PRM can be used to help organizations conduct a gap analysis of existing practices. The results of a gap analysis can be used to prioritize and track SwA implementation efforts. The Assurance PRM addresses assurance from executive to developer. |
| BSIMM2: The Building Security In Maturity Model | [http://bsimm2.com/](http://bsimm2.com/) | Pronounced “bee simm” was created by observing and analyzing real-world data from thirty leading software security initiatives. The BSIMM can help you determine how your organization compares to other real-world software security initiatives and what steps can be taken to make your approach more effective. |
| CERT Resilience Management Model | [http://www.cert.org/resilience/rmm.html](http://www.cert.org/resilience/rmm.html) | It has two primary objectives:  
1. Establish the convergence of operational risk and resilience management activities such as security, business continuity, and aspects of IT operations management into a single model.  
2. Apply a process improvement approach to operational resilience management through the definition and application of a capability level scale that expresses increasing levels of process improvement. |
# Workforce Credentials

<table>
<thead>
<tr>
<th>Certification Authority</th>
<th>SwA Relevant Certificates</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC-Council</td>
<td>EC-Council Certified Secure Programmer (ECSP)</td>
<td><a href="http://www.eccouncil.org/certification.htm">http://www.eccouncil.org/certification.htm</a></td>
</tr>
<tr>
<td></td>
<td>(Technologies Covered: C/C++, Java, .Net, PHP, SQL)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Certified Secure Application Developer (CSAD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Certified Ethical Hacker (CEH)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Licensed Penetration Tester (LPT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIAC Secure Software Programmer - Java (GSSP-JAVA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIAC Web Application Penetration Tester (GWAPT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIAC Certified Penetration Tester (GPEN)</td>
<td></td>
</tr>
<tr>
<td>IEEE Computer Society</td>
<td>Certified Software Development Professional (CSDP)</td>
<td><a href="http://www.computer.org/portal/web/certification">http://www.computer.org/portal/web/certification</a></td>
</tr>
<tr>
<td>(ISC)²</td>
<td>Certified Secure Software Lifecycle Professional (CSSLP)</td>
<td><a href="http://www.isc2.org/csslp-certification.aspx">http://www.isc2.org/csslp-certification.aspx</a></td>
</tr>
</tbody>
</table>
Job Roles

• What kind of jobs can I get?
  – Jobs and career planning
    • http://www.sans.org/20coolestcareers

#18 - Security-savvy Software Developer*
"Kool, because this is VERY rare."

Job Description
The security-savvy software developer leads all developers in the creation of secure software, implementing secure programming techniques that are free from logical design and technical implementation flaws. This expert is ultimately responsible for ensuring customer software is free from vulnerabilities that can be exploited by an attacker.

#2 - System, Network, and/or Web Penetration Tester* - Top Gun Job
"You can be a hacker, but do it legally and get paid a lot of money!"
Role Descriptions

» **Cyber Software Assurance Developer/Integrator**
  - Experience with applying security activities within SDLC
  - Experience with security, including CSSLP, CISSP and SANS secure programming assessments
  - Experience with security standards, including SSE-CMM, NIST SPs, ISO 15408
  - Common Criteria, or client-specific software assurance guides. (Also see the section on “Standards of Practice”)

» **Software Assurance Engineer**
  - Provide technical leadership in all aspects of software assurance and computer systems engineering support
  - Lead and actively participate in the evaluation and analyses of activities related to all phases of the secure software life cycle from initial planning, requirements definition, design and development through integrated system testing and sustaining operations.
  - Responsibilities will also include the support of a wide range of technical and programmatic activities for program offices, including leading the review and assessment of software system architecture; system requirements and their allocation to lower level specifications; design, code and test activities; trade studies; COTS/GOTS products; reuse software; test tools; simulators; software verification and validation (V&V); and system test and integration. Support independent review efforts in analyzing and assessing system software and related development and testing activities.
Information Security Systems & Software Development Professional (ISSSDP)

The Information Security Systems and Software Development Professional is responsible for secure design, development, testing, integration, implementation, sustainment, and/or documentation of software applications (web based and non-web) following formal secure systems development lifecycle processes and using security engineering principles.

The following professional requirements form part of a broad Federal Government effort to identify and describe roles.

<table>
<thead>
<tr>
<th>Table 8– Credentials, Competencies and Skills by Performance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Entry Level</strong></td>
</tr>
<tr>
<td>Software Development</td>
</tr>
<tr>
<td>Written &amp; Oral Communication</td>
</tr>
<tr>
<td>Creative Problem Solving</td>
</tr>
<tr>
<td>Information Security/Assurance</td>
</tr>
<tr>
<td>Critical Thinking and Analytical Skills</td>
</tr>
<tr>
<td>Project/Program Management</td>
</tr>
<tr>
<td>Leadership &amp; People Management</td>
</tr>
<tr>
<td>Suggested Credentials in:</td>
</tr>
<tr>
<td>▪ Computer science/engineering</td>
</tr>
<tr>
<td>▪ Database/information management</td>
</tr>
<tr>
<td>▪ Information assurance/security</td>
</tr>
<tr>
<td>▪ Software assurance/security</td>
</tr>
<tr>
<td>▪ Information systems management</td>
</tr>
</tbody>
</table>
Got Content?

• The pocket guide is a “work in progress”

• Plenty of opportunity to contribute content

• Join the Effort!
  – Your comments, suggestions, criticism/praise are all very welcome
Where to find the PocketGuide?

• https://buildsecurityin.us-cert.gov/swa/pocket_guide_series.html

• And many others...
Software Assurance Pocket Guide Series

The SWA Pocket Guide Series comprises free, downloadable documents on software assurance in acquisition and outsourcing, software assurance in development, the software assurance life cycle, and software assurance measurement and information needs. SWA Pocket Guides are developed collaboratively by the SWA Forum and SWA Working Groups, which function as a stakeholder community that welcomes additional participation in advancing and refining software security. Your input on these documents is welcome; please use the feedback form. For general inquiries, please email Software.Assurance@dhs.gov.

- **SWA in Acquisition and Outsourcing**
  - Software Assurance in Acquisition and Contract Language
  - Software Supply Chain Risk Management and Due Diligence

- **SWA in Development**
  - Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses
  - Software Security Testing
  - Requirements and Analysis for Secure Software
  - Architecture and Design Considerations for Secure Software
  - Secure Coding

- **SWA Life Cycle**
  - Software Assurance in Education, Training & Certification

- **Future SWA Pocket Guides**

---

**Software Assurance in Education, Training and Certification**

*Life Cycle Support Volume I - (Version 2.1, March 1, 2011)*

Current events related to cybersecurity encourage a fundamental shift in the way we think about educating and training a workforce prepared to address security issues in all phases of a software system. Software assurance education and training is aimed to ensure adequate coverage of requisite knowledge areas in contributing disciplines such as software engineering (including its many subdisciplines), systems engineering, project management, etc., to identify and acquire competencies associated with secure software. The primary audiences for this pocket guide are educators and trainers who can use this guide to identify resources to supplement their efforts as well as identify strategies to inject software assurance related topics into the existing education and training programs.

8.5" x 11" version PDF file

**Future SWA Pocket Guides**
Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses
Development Volume II - (Version 1.3 May 24, 2009)
Common Weakness Enumeration (CWE) provides a standard means for understanding residual risks; enabling more informed decision-making by suppliers and consumers about the security of software. The 2009 CWE/SANS Top 25 Most Dangerous Programming Errors is a list of the most egregious programming errors that can lead to serious exploitable software vulnerabilities. The main goal for the Top 25 list is to stop vulnerabilities at the source by educating programmers on how to eliminate the most egregious programming errors before software is shipped. The list could be used as a tool for education and awareness that helps programmers prevent the kinds of vulnerabilities that plague the software industry. Software consumers could use the same list to help them to ask for more secure software, while software managers and CIOs could use the Top 25 list as a measuring stick of progress in their efforts to secure their software. This volume of pocket guide links the CWE's with the common attacks that exploit these weaknesses, the resulting mission and business risks. It provides recommended practices for preventing the exploits in software.

Software Security Testing
Development Volume III - (Version 0.7 May 10, 2010)
Software security testing validates the secure implementation of a product thus reducing the likelihood of security flaws being released and discovered by customers or malicious users. The goal is not to "test in security," but to validate the robustness and security of the software products prior to making them available to customers and to prevent security vulnerabilities from ever entering the software. This volume of the pocket guide describes the most effective security testing techniques, their strengths and weaknesses, and when to apply them during the Software Development Life Cycle.

Requirements and Analysis for Secure Software
Development Volume IV - (Version 1.0, October 5, 2009)
Comprehensive requirements are critical for successful system development, but all too often, requirements fail to explicitly consider security. As a result, systems meet the functionality but are rarely safe and consequently are the target of attacks. Systems which carefully document security requirements reduce the likelihood of successful attacks. Security requirements include functions that implement a security policy such as areas of access control, identification, authentication and
SwA in Development
- Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses
- Software Security Testing
- Requirements and Analysis for Secure Software
- Architecture and Design Considerations for Secure Software
- Secure Coding

SwA Life Cycle
- Software Assurance in Education, Training & Certification

Future SwA Pocket Guides

SwA in Acquisition and Outsourcing

Software Assurance in Acquisition and Contract Language
Acquisition and Outsourcing Volume I - (Version 1.1 July 31, 2009)
Integrating software security in the acquisition life cycle promotes the acquisition of secure software. This volume of the pocket guide includes samples SwA Request for Proposal (RFP)/Contract language. Buyers and evaluators of software and suppliers can gain security risk-based insight. They can put suppliers on notice that consumers are concerned about software security and the risks to their organizations that are attributable to exploitable software.

8.5" x 11" version PDF File

Software Supply Chain Risk Management and Due Diligence
Acquisition and Outsourcing Volume II - (Version 1.2 June 16, 2009)
Software security enhanced due-diligence is a critical element of software supply chain risk management. The focus of the volume is to increase awareness for the need to include software assurance and identify best practices in the acquisition of software. Due-diligence involves taking reasonable steps to ensure that software or a software-intensive system not only meets functional and technical requirements, but also addresses software assurance concerns. Buyers and evaluators of software and services can gain security risk-based insight. They can put suppliers on notice that consumers are concerned about software security and the risks to their organizations that are attributable to exploitable software.

8.5" x 11" version PDF File
Find me

• Robin A. Gandhi, Ph.D.
  Assistant Professor of Information Assurance
  University of Nebraska at Omaha

rgandhi@unomaha.edu

Voice: (402) 554 3363,  Fax: (402) 554-3284

http://faculty.ist.unomaha.edu/rgandhi
Acknowledgement

• Joe Jarzombek for giving me the opportunity to lead this effort
• Members of the SwA WG on Education and Training for insightful comments, reviews and content (Dan, Carol, Nancy, Art)
• Susan Morris, Walter Houser, Dominick Chiriyan
• And many others...
Bonus Slides
Why Johnny Can’t write secure code?

• *Johnny, avoid these weaknesses…. Period!*
  – Common Weaknesses Enumeration (CWE)

• *Johnny...learn from your mistakes*
  – Common Vulnerabilities and Exposures (CVE)

• *Johnny...these are the ways of the bad guys*
  – Common Attack Patterns Enumeration and Classification (CAPEC)

• *Johnny...these are ways to develop secure code*
  – CERT secure coding guidelines
Poor Johnny!

- 45,000+ CVE Vulnerabilities
- CWE 650+ Weaknesses 1000+ Pages
- Countless Do’s and Don’ts

CAPEC 300+ Attack Patterns
Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories

Me

Harvey Siy

Yan Wu
The Paradox we face!
Tangling of information in the CWE

• **CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer**
  
  – The software performs operations on a *memory buffer*, but it can read from or write to a memory location that is outside of the intended boundary of the *buffer*.
  
  – Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data. As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.
Tangling of information in the CWE

- **CWE-120: Buffer Copy without Checking Size of Input** ('Classic Buffer Overflow')

  - The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
  
  - A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer.

  - Buffer overflows often can be used to execute arbitrary code...

  - Buffer overflows generally lead to crashes
Buffer Overflow

**WEAKNESS**

- **ACCESS AND OUT-OF-BOUNDS WRITE #125, #126, #127, #786**
- **ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786**
- **FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119**

**CONSEQUENCES**

- **WRITE-WHAT-WHERE CONDITION #123**
- **UNCONTROLLED MEMORY ALLOCATION #789**
- **PART-OF**
  - **BUFFER #119**
  - **MEMORY-BUFFER #119**
- **PART-OF**
  - **INDEXABLE-RESOURCE #118**

**SOFTWARE-FAULT**

- **WRAP-AROUND ERROR #128**
- **INTEGER OVERFLOW #190 #680**
- **INTEGER UNDERFLOW #191**
- **POINTER ERRORS #467 #468**
- **IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY #130**

**CAN-PRECEDE**

- **SOFTWARE-FAULT**
  - **SIGN ERRORS #194 #195 #196**
  - **IMPROPER-INPUT-VALIDATION #20**
  - **INCORRECT-BUFFER-SIZE-CALCULATION #131**

**RESOURCES/LOCATION**

- **STACK-BASED #121**
- **STATIC #129**
- **HEAP-BASED #122**
- **INDEX (POINTER #466 INTEGER #129)**

**CAN-PRECEDE**

- **SOFTWARE-FAULT**
  - **INTEGER COERCION ERROR #192**
  - **MISSING INITIALIZATION #456**
  - **API ABUSE #227**
  - **USE OF DANGEROUS FUNCTIONS #242**
  - **IMPROPER USE OF FREED MEMORY #415 #416**

**OCCURS-IN**

- **MISSING INITIALIZATION #456**
- **API ABUSE #227**
- **USE OF DANGEROUS FUNCTIONS #242**
- **IMPROPER USE OF FREED MEMORY #415 #416**
  - **BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ("CLASSIC BUFFER OVERFLOW") #120**

**IS-A**

- **SOFTWARE-FAULT**
- **INTEGER UNDERFLOW #191**
- **INTEGER OVERFLOW #190 #680**
- **WRAP-AROUND ERROR #128**
- **POINTER ERRORS #467 #468**
- **IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY #130**

**IS-A**

- **SOFTWARE-FAULT**
- **SIGN ERRORS #194 #195 #196**
- **IMPROPER-INPUT-VALIDATION #20**
- **INCORRECT-BUFFER-SIZE-CALCULATION #131**

**IS-A**

- **SOFTWARE-FAULT**
- **INTEGER UNDERFLOW #191**
- **INTEGER OVERFLOW #190 #680**
- **WRAP-AROUND ERROR #128**
- **POINTER ERRORS #467 #468**
- **IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY #130**

**IS-A**

- **SOFTWARE-FAULT**
- **ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124**
- **FAILURe TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119**
- **IMPROPER-ACCESS-OF-INDEXABLE-RESOURCE #118**
**Buffer Overflow Semantic template**

**WEAKNESS**

- **OFFSET-ONE #193**
- **INACCESSIBLE-RESOURCE #118**
- **ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786**
- **FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119**
- **IMPROPER ACCESS OF INDEXABLE-RESOURCE #118**
- **INCORRECT-CALCULATION #682**
- **WRAP-AROUND ERROR #128**
- **POINTER ERRORS #467 #468**
- **IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY #130**
- **IMPROPER HANDLING OF OUT-OF-BOUNDS VALUE OUTSIDE OF EXPECTED RANGE #466**
- **IMPROPER NULL TERMINATION #170**
- **IMPROPER VALIDATION OF ARRAY INDEX #129 #789**

**CONSEQUENCES**

- **WRITE-WHAT-WHERE CONDITION #123**
- **WRONG TYPE USE #192**
- **UNCONTROLLED MEMORY ALLOCATION #789**

**CVE-2010-1773**

**CVE Description**: *Off-by-one error* in the toAlphabetic function in rendering/RenderListMarker.cpp

**Change Log Issue Description**: The math was slightly off here, and we wound up trying to access an array at index -1 in some cases

**Change Log Fix Description**: We need to decrement `numberShadow` rather than subtracting one from the result of the modulo operation

**Code Change for Fix**: Line 105 decrement `--numberShadow;` and remove the subtraction of one in Line 106 `(sequence[numberShadow % sequenceSize];)`

**CVE Description**: ….trying to access an array at index -1 in some cases

**Change Log Issue Description**: ….trying to access an array at index -1 in some cases

**Change Log Fix Description**: We need to decrement `numberShadow` rather than subtracting one from the result of the modulo operation

**Code Change for Fix**: Line 105 decrement `--numberShadow;` and remove the subtraction of one in Line 106 `(sequence[numberShadow % sequenceSize];)`

**CVE Description**: ….trying to access an array at index -1 in some cases

**Change Log Issue Description**: ….trying to access an array at index -1 in some cases

**Change Log Fix Description**: We need to decrement `numberShadow` rather than subtracting one from the result of the modulo operation

**Code Change for Fix**: Line 105 decrement `--numberShadow;` and remove the subtraction of one in Line 106 `(sequence[numberShadow % sequenceSize];)`

**CVE Description**: …allows remote attackers to obtain sensitive information...

**CVE Description**: ….allows remote attackers to obtain sensitive information...

**CVE-2010-1773 IS-A Buffer Overflow Semantic template**
Experiment

• The scenario...
  – A newbie programmer or occasional contributor to open source project
    • How much effort does it take to study a vulnerability and summarize lessons learned?

• 30 Computer Science students from a senior-level undergraduate Software Engineering course.
  – None to more than 5 years
  – No prior knowledge of semantic templates
• $H_{10}$:
  – There is no reduction in completion time for subjects who use semantic templates compared to those who do not.

• $H_{20}$:
  – There is no improvement in accuracy of understanding of vulnerabilities for subjects who use semantic templates compared to those who do not.
Variables

• The experiment manipulated these *independent variables*:
  – **Group** - refers to the group assigned (1 or 2).
  – **Round** - refers to the experiment round (1 or 2).

• **Vulnerability ID** - the vulnerability under study (1-1, 1-2, 1-3, 2-1, 2-2, 2-3).
  – These self-reported *subject variables* were collected:
    • Programming skill level
    • *Reading comprehension and writing skill levels* - ability to read and write technical English documents.
Variables

• Dependent variables:
  – Time to complete assignment
  – CWE identification accuracy
  – Fault identification accuracy
    • a score (scale of 1-5) on the accuracy of the identification of the software fault that led to the vulnerability
  – Failure identification accuracy
    • a score (scale of 1-5) on the accuracy of the description of the nature of the vulnerability (the manifested problem, the resources impacted and the consequences)
Initial Results and Findings

Table 1: p-values of one-tailed t-tests for Time data

| Round 1 | (1-1) 0.3627 | (1-2) 0.5855 | (1-3) 0.1516 |
| Round 2 | (2-1) **0.0001** | (2-2) **0.0030** | (2-3) **0.0015** |

p-values of one-tailed t-tests for CWE precision

| Round 1 | (1-1) 0.9281 | (1-2) 0.9957 | (1-3) 0.5344 |
| Round 2 | (2-1) 0.1840 | (2-2) 0.6023 | (2-3) 0.0891 |

Table 1: p-values of one-tailed t-tests for CWE recall

| Round 1 | (1-1) 0.0683 | (1-2) 0.9481 | (1-3) 0.2286 |
| Round 2 | (2-1) **0.0141** | (2-2) **0.0093** | (2-3) **0.0021** |

Time to completion (minutes) per vulnerability
Future Work

• Integrate with existing static and dynamic analysis tools to enhance reporting capabilities
  – Provide layers of guidance to a developer upon detection of a software flaw
  – Organize and retrieve knowledge of past vulnerabilities
  – Verify patch submissions

• Investigate project/developer specific coding errors and vulnerability fix patterns

• Other usage scenarios in the SDLC
This research is funded in part by Department of Defense (DoD)/Air Force Office of Scientific Research (AFOSR), NSF Award Number FA9550-07-1-0499, under the title “High Assurance Software”
Thank you for your Attention