Software Assurance: Enabling Enterprise Resilience through Security Automation and Software Supply Chain Risk Management

Joe Jarzombek, PMP, CSSLP
Director for Software Assurance
Cyber Security & Communications
US Department of Homeland Security
Public/Private Collaboration Efforts for Security Automation and Software Supply Chain Risk Management

Next SwA Forum sessions: 5-7 March 2013 at NIST in Gaithersburg MD
Interdependencies Between Physical & Cyber – Convergence of Safety, Security and Resilience Considerations

In an era riddled with asymmetric cyber attacks, claims about system reliability and safety must include provisions for built-in security of the enabling software.
Interdependencies Between Physical & Cyber Infrastructures: Convergence of Safety, Security and Dependability Considerations

-- Need for secure/resilient software applications

Software is a high risk component
Today Everything’s Connected

Your System is attackable…

When this Other System gets subverted through an un-patched vulnerability, a mis-configuration, or an application weakness…
Exploitable Software Weaknesses (CWEs) are sources for future Zero-Day Attacks

- Cross-site Scripting (XSS) Attack (CAPEC-86)
- Improper Neutralization of Input During Web Page Generation (CWE-79)
- SQL Injection Attack (CAPEC-66)
- Improper Neutralization of Special Elements used in an SQL Command (CWE-89)
Security is a Requisite Quality Attribute: Vulnerable Software Enables Exploitation

- Rather than attempt to break or defeat network or system security, hackers are opting to target application software to circumvent security controls.
  - 75% of hacks occurred at application level
    - “90% of software attacks were aimed at application layer” (Gartner & Symantec, June 2006)
  - most exploitable software vulnerabilities are attributable to non-secure coding practices (and not identified in testing).
- Functional correctness must be exhibited even when software is subjected to abnormal and hostile conditions.

In an era riddled with asymmetric cyber attacks, claims about system reliability, integrity & safety must include provisions for built-in security of the enabling software.
Software Assurance Addresses Exploitable Software:
Outcomes of non-secure practices and/or malicious intent

Exploitation potential of vulnerability is independent of “intent”

Defects

EXPLOITABLE SOFTWARE

Unintentional Vulnerabilities

Intentional Vulnerabilities

Malware

‘High quality’ can reduce security flaws attributable to defects; yet traditional S/W quality assurance does not address intentional malicious behavior in software

Software Assurance (SwA) is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.*

From CNSS Instruction 4009 “National Information Assurance Glossary” (26APR2010)
IT/software security risk landscape is a convergence between “defense in depth” and “defense in breadth”

Enterprise Risk Management and Governance are security motivators

Acquisition could be considered the beginning of the lifecycle; more than development

“In the digital age, sovereignty is demarcated not by territorial frontiers but by supply chains.”
– Dan Geer, CISO In-Q-Tel

Software Assurance provides a focus for:
-- Secure Software Components,
-- Security in the Software Life Cycle,
-- Software Security in Services, and
-- Software Supply Chain Risk Management
Risk Management (Enterprise ↔ Project): Shared Processes & Practices ↔ Different Focuses

- **Enterprise-Level:**
  - Regulatory compliance
  - Changing threat environment
  - Business Case

- **Program/Project-Level:**
  - Cost
  - Schedule
  - Performance

Who makes risk decisions?
Who “owns” residual risk from tainted/counterfeit products?

* “Tainted” products are those that are corrupted with malware, or exploitable weaknesses & vulnerabilities that put users at risk
Even after vulnerabilities are discovered and patches made available, many developers use the flawed, non-patched version of reused components.

Who makes risk decisions?

Who inherits the residual risk?

Who ‘owns’ the residual risk attributable to exploitable software?

Source: Maximizing Benefits and Mitigating Risks of Open Source Components in Application Development, by Sonatype
Assessment Challenges of Multiple Platforms

Operating System + CPU + Compiler + Libraries

- Windows
- Linux
- Solaris
- Greenhills
- AMD
- Intel
- Sun Microsystems
- GCC
- MS Visual Studio
- IAR
- Sun CC
- Greenhills
- Diab
- OS libraries
- C library
- Networking
- Thread libraries
- Crypto
- 32-bit
- 64-bit
- compiler flags
- multiple versions
- multiple distros

© GrammaTech, Inc. 2012. Proprietary; Publicly shared at Software Assurance working Group session on 28 Nov 2012
Challenges in Mitigating Risks Attributable to Exploitable Software and Supply Chains

- Complexity hampers ability to determine and predict code behavior; so any “assurance” claims for security/safety-critical applications are limited.

- Without adequate diagnostic capabilities and commonly recognized standards from which to:
  - discern product assurance;
  - benchmark process capabilities, and
  - assert claims about the assurance of products, systems and services,

- “provenance and pedigree of supply chain actors” become a more dominant consideration for security/safety-critical applications:
  - Enterprises and Users lack requisite transparency for more informed decision-making for mitigating risks;
  - Favoring domestic suppliers does not necessarily address ‘assurance’ in terms of capabilities to deliver secure/safe components, systems or software-reliant services.
Several needs arise:

- Need internationally recognized standards to support security automation and processes to provide transparency for more informed decision-making for mitigating enterprise risks.
- Need ‘Assurance’ to be explicitly addressed in standards & capability benchmarking models for organizations involved with security/safety-critical applications.
- Need more comprehensive diagnostic capabilities to provide sufficient evidence that “code behavior” can be well understood to not possess exploitable or malicious constructs.
- Need rating schemes for software products and supplier capabilities.
Mitigating Risks Attributable to Exploitable Software and Supply Chains

Enterprises seek comprehensive capabilities to:

► Avoid accepting software with **MALWARE** pre-installed.

► Determine that no publicly reported **VULNERABILITIES** remain in code prior to operational acceptance, and that future discoveries of common vulnerabilities and exposures can be quickly patched.

► Determine that exploitable software **WEAKNESSES** that put the users most at risk are mitigated prior to operational acceptance or after put into use.
DHS Software Assurance Program Overview

- Program established in response to the National Strategy to Secure Cyberspace - Action/Recommendation 2-14:

  “DHS will facilitate a national public-private effort to promulgate best practices and methodologies that promote integrity, security, and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, malicious code, or trap doors that could be introduced during development.”

- DHS Program goals promote the security and resilience of software across the development, acquisition, and operational life cycle

- DHS Software Assurance (SwA) program is scoped to address:
  - **Trustworthiness** - No exploitable vulnerabilities or malicious logic exist in the software, either intentionally or unintentionally inserted,
  - **Dependability (Correct and Predictable Execution)** - Justifiable confidence that software, when executed, functions as intended,
  - **Survivability** - If compromised, damage to the software will be minimized; it will recover quickly to an acceptable level of operating capacity; it’s ‘rugged’;
  - **Conformance** – Planned, systematic set of multi-disciplinary activities that ensure processes/products conform to requirements, standards/procedures.

See Wikipedia.org for “Software Assurance” - CNSS Instruction No. 4009, "National Information Assurance Glossary," Revised 2006, defines Software Assurance as: "the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner".
DHS CS&C Software Assurance (SwA) Program

Advances security and resilience of software throughout the lifecycle; focuses on reducing exploitable software weaknesses and addresses means to improve capabilities that routinely develop, acquire, and deploy resilient software.

• Serves as a focal point for interagency public-private collaboration to enhance development and acquisition processes, capability benchmarking and rating schemes to address software security needs.
  – Hosts interagency Software Assurance Forums, working groups and training to provide public-private collaboration in advancing software security and providing publicly available resources.
  – Provides collaboratively developed, peer-reviewed information resources on Software Assurance, via journals, guides & on-line resources suitable for use in education, training, and process improvement.
  – Provides input and criteria for leveraging international standards and maturity models used for process improvement and capability benchmarking of software suppliers and acquisition organizations.

• Enables software security automation and measurement capabilities through use of common indexing and reporting capabilities for malware, exploitable software weaknesses, cyber indicators and attacks which target software.
  – Collaborates with national & international standards organizations and industry to create standards, metrics and certification mechanisms from which products and tools could be qualified for software security verification.
  – Manages programs for Malware Attribute Enumeration Classification (MAEC), Common Weakness Enumeration (CWE), Common Attack Pattern (CAPEC) & Cyber Observable eXpression (CybOX).
  – Manages programs for Common Vulnerabilities & Exposures (CVE) and Open Vulnerability & Assessment Language (OVAL) that provide information feeds for continuous monitoring, security content automation, vulnerability databases, and security/threat alerts from many organizations.
# Software Assurance Forum & Working Groups*

... encourage the production, evaluation and acquisition of more secure and resilient software through targeting:

<table>
<thead>
<tr>
<th>People</th>
<th>Processes</th>
<th>Technology</th>
<th>Acquisition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers and users education &amp; training</td>
<td>Sound practices, standards, &amp; practical guidelines for secure software development</td>
<td>Security test criteria, measurement, diagnostic tools, common languages &amp; enumerations, SwA Research &amp; Development</td>
<td>Software security improvements through due-diligence questions, specs and guidelines for acquisitions/outsourcing</td>
</tr>
</tbody>
</table>

## Products and Contributions

- **Build Security In** - https://buildsecurityin.us-cert.gov and SwA community resources & info clearinghouse
- **SwA Common Body of Knowledge (CBK) & Glossary**
- **Organization of SwSys Security Principles/Guidelines**
- **SwA Developers' Guide on Security-Enhancing SDLC**
- **SwA Curriculum Project: Masters and Undergraduate**
- **Software Security Assurance State of the Art Report**
- **Systems Assurance Guide (via DoD and NDIA)**
- **SwA-related standards – ISO/IEC JTC1 SC7/27/22, IEEE CS, OMG, TOG, & CMM-based Assurance**
- **Practical Measurement Framework for SwA/InfoSec**
- **Making the Business Case for Software Assurance**
- **SwA Metrics & Tool Evaluation (with NIST)**
- **SwA Ecosystem w/ DoD, NSA, NIST, OMG & TOG**
- **NIST Special Pub 500 Series on SwA Tools**
- **Security Automation Enumerations & Languages (CVE, OVAL, CWE, CAPEC, MAEC, CybOX, STIX), scoring systems, and risk analysis capabilities**
- **SwA in Acquisition: Mitigating Risks to Enterprise**
- **Software Project Management for SwA SOAR**

*SwA Forum is part of Cross-Sector Cyber Security Working Group (CSCSWG) established under auspices of the Critical Infrastructure Partnership Advisory Council (CIPAC) that provides legal framework for participation.*
DHS Software Assurance (SwA) Outreach & Awareness

- Co-sponsor SwA working group sessions, semi-annual SwA Forum, for government, academia, and industry to facilitate ongoing public-private collaboration.

- Provide SwA presentations, workshops, and tracks at conferences

- Co-sponsor SwA issues of CROSSTALK to “spread the word”
  - Sep 2008 issue on “Application Security”
  - Mar/Apr 2009 issue on “Reinforcing Good Practices”
  - Sep/Oct 2009 issue on “Resilient Software”
  - Mar/Apr 2010 issue on “System Assurance”
  - Sep/Oct 2010 issue on “Game Changing Tools & Practices”
  - Mar/Apr 2011 issue on “Rugged Software”
  - Sep/Oct 2011 issue on “Protecting against Predatory Practices”
  - Mar/Apr 2012 issue on “Securing a Mobile World”
  - Sep/Oct 2012 issue on “Resilient Cyber Ecosystem”
  - Mar/Apr 2013 issue on “Supply Chain Risk Management”

- Collaborate with standards organizations, consortiums, professional societies, education/training initiatives in promoting SwA

- Provide free SwA resources via “BuildSecurityIn” website to promote secure development methodologies (since Oct 05)

- Host SwA Community Resources & Information Clearinghouse via https://buildsecurityin.us-cert.gov/SwA (since Dec 07)
BSI https://buildsecurityin.us-cert.gov focuses on making Software Security a normal part of Software Engineering.

SwA Community Resources and Information Clearinghouse (CRIC)

https://buildsecurityin.us-cert.gov/swa/ focuses on all contributing disciplines, practices and methodologies that advance risk mitigation efforts to enable greater resilience of software/cyber assets.

The SwA CRIC provides a primary resource for SwA Working Groups.

Where applicable, SwA CRIC & BSI provide relevant links to each other.
Life-Cycle Standards View Categories (ISO/IEC 15288 and 12207)

**Organization**
- Governance Processes
  - Strategy and policy
  - Enterprise risk management
    - Compliance
    - Business case
  - Supply Chain Management
- Project-Enabling Processes
  - Life Cycle Model Management
  - Infrastructure Management
    - SwA ecosystem
    - Enumerations, languages, and repositories
  - Project Portfolio Management
  - Human Resource Management
    - SwA education
    - SwA certification and training
    - Recruitment
  - Quality Management
- Agreement Processes
  - Acquisition
    - Outsourcing
    - Agreements
    - Risk-based due diligence
    - Supplier assessment
  - Supply

**Project**
- Project Management Processes
  - Project Planning
  - Project Assessment and Control
    - Assurance case management
- Project Support Processes
  - Decision Management
  - Risk Management
    - Threat Assessment
  - Configuration Management
  - Information Management
  - Measurement

**Engineering**
- Technical Processes
  - Stakeholder Requirements Definition
  - Requirements Analysis
    - Attack modeling (misuse and abuse cases)
    - Data and information classification
    - Risk-based derived requirements
    - Sw security requirements
  - Architectural Design
    - Secure Sw architectural design
    - Risk-based architectural analysis
    - Secure Sw detailed design and analysis
  - Implementation
    - Secure coding and Sw construction
    - Security code review and static analysis
    - Formal methods
  - Integration
    - Sw component integration
    - Risk analysis of Sw reuse components
  - Verification & Validation
    - Risk-based test planning
    - Security-enhanced test and evaluation
      - Dynamic and static code analysis
      - Penetration testing
      - Independent test and certification
  - Transition
    - Secure distribution and delivery
    - Secure software environment (secure configuration, application monitoring, code signing, etc)
- Operations and Sustainment
  - Operation
    - Incident handling and response
  - Maintenance
    - Defect tracking and remediation
    - Vulnerability and patch management
    - Version control and management
  - Disposal

**Software Reuse Processes**
- Domain Engineering
- Reuse Asset Management
- Reuse Program Management

**Software Support Processes**
- Sw Documentation Management
- Sw Quality Assurance
- Sw Configuration Management
- Sw Verification & Sw Validation
- Sw Review
- Sw Audit
- Sw Problem Resolution
Software Assurance (SwA) Pocket Guide Series

SwA in Acquisition & Outsourcing
• Software Assurance in Acquisition and Contract Language
• Software Supply Chain Risk Management and Due-Diligence

SwA in Development *
• Risk-based Software Security Testing
• Requirements and Analysis for Secure Software
• Architecture and Design Considerations for Secure Software
• Secure Coding and Software Construction
• Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses
  * Include questions to ask developers

SwA Life Cycle Support
• SwA in Education, Training and Certification

SwA Pocket Guides and SwA-related documents are collaboratively developed with peer review; they are subject to update and are freely available for download via the DHS Software Assurance Community Resources and Information Clearinghouse at https://buildsecurityin.us-cert.gov/swa (see SwA Resources)
Objectives for SCRM & SwA in Acquisition

- we need “systems-of-systems” or “enterprise systems” thinking for risk management (building on 800-39 and 800-64)
- IT Baselines for SCRM & SwA should build on 800-53
Applying Relevant Guidance to SCRM

Industry and NIST Documents as well as other inputs that will shape the development of NIST SP800-161:

- 800-39
- 800-30rev1
- 800-53rev4
- FIPS 199
- NIST IR on SCRM
- SCRM Workshop Feedback
- Inputs from other organizations: Open Group, OMG, ISO/IEC JTC1, SafeCode, etc.
Supply Chain Risk Management

◆ SCRM is an enterprise activity that impacts organizational, mission, and operational layers. Must ensure coverage of all layers within the risk management structure.

◆ SCRM SP will be based on risk management and its process (NIST SP 800-39)
  o Frame (F)
  o Assess (A)
  o Respond (R)
  o Monitor (M)

◆ Supply chain risk management will incorporate
  o Criticality Analysis; FIPS 199 (only high & moderate necessitate SCRM)
  o Threat
  o Vulnerabilities
INTRODUCTION

1.1 PURPOSE AND APPLICABILITY
1.2 TARGET AUDIENCE
1.3 RELATIONSHIP TO OTHER PUBLICATIONS
1.4 ORGANIZATIONAL RESPONSIBILITIES
1.5 ORGANIZATION OF THIS SPECIAL PUBLICATION

THE FUNDAMENTALS

2.1 SUPPLY CHAIN RISK MANAGEMENT
2.2 ASSURANCE AND TRUSTWORTHINESS

THE PROCESS

3.1 FRAMING SUPPLY CHAIN CONCERNS
3.2 ASSESSING SUPPLY CHAIN CONCERNS
3.3 RESPONDING TO THE SUPPLY CHAIN CONCERNS
   - SCRM GUIDANCE / BASELINES
   - TAILORING BASELINE SCRM GUIDANCE
   - DOCUMENTING THE GUIDANCE SELECTION PROCESS
   - DOCUMENTING THE GUIDANCE FOR CONTINUOUS MONITORING
3.4 MONITORING FOR SUPPLY CHAIN ASSURANCE
   - DOCUMENTING THE GUIDANCE FOR CONTINUOUS MONITORING
The Fundamentals

◆ SUPPLY CHAIN RISK MANAGEMENT
  Defines the structure and elements of SCRM
  o Risk Management Pyramid
    (using the Frame, Assess, Respond, Monitor -FARM Process)
  o Relationship between Organizational, Business and Operational layers addressed with respect to SCRM
  o Scope of baseline guidance – what the baseline guidance will cover
  o How to verify if guidance is sufficient to address SCRM for the specific agency

◆ ASSURANCE AND TRUSTWORTHINESS
  o Address the criticality and FIPS 199 processes
  o Discuss threats – both assumed and new
    • How threats may be scoped and how they tie into criticality & influence SCRM
  o Discuss vulnerabilities – both assumed and new
    • How vulnerabilities may be scoped and how they tie into criticality & influence SCRM
  o Describe users/acquirers and their relationship with suppliers
The Process

This Section (Chapter 3) walks the acquirer through the SCRM implementation process using FARM.

Frame (F):

Provides the acquirer with the foundational structure to address the various threats and vulnerabilities to manage SCRM

- Defines the Scope including criticality, threat and vulnerability landscape of the supply chain system/component
- Discusses the various organizational and business/mission requirements and constraints that may apply
- List of information sources for acquirer to reach out to for obtaining relevant data/information for supply chain risk assessment.
The Process

Assess (A):
Provides the structure for agency specific assessment where organization specific SCRM context can be defined enabling the assessment of specific threat or vulnerability condition for impact.

- Define the specific threats and vulnerabilities within the context of the organizational threat landscape and organizational vulnerability landscape.
- Identify the various non-compliance or non-conformance issues that may contribute to the supply chain risk.
- Identify the stakeholders potentially impacted by the vulnerability or threat.
- Assess the overall impact of the vulnerability or threat in the context of the specific supply chain issue.
The Process

Respond (R):

Discusses the SCRM guidance as it pertains to the organizational, business, and operational layers of landscape for addressing SCRM.

- **SCRM Guidance/Baseline**
  Lists SCRM reference documents and guidance that can be leveraged by the acquirer.

- **Tailoring SCRM Guidance/Baseline**
  Provides a more tailored approach for choosing the relevant guidance that may pertain to that specific organizational needs.

- **Documenting the Guidance Selection Process**
  Discusses the traceability and reasoning for specific SCRM guidance for a specific agency and situation. There is a tie to existing 800-53 controls where relevant and any enhanced measures required for managing risk.
  (Use applicable security controls/assurance overlays)
Monitor (M):

This section discusses guidance for monitoring or continuous monitoring as it applies to SCRM.

- Review the various guidance and controls schema for potential use in a continuous monitoring program.
  - The guidance and controls need to either be repeatable and/or automated for continuous monitoring.

Note: This section may be a phase one view of continuous monitoring - can evolve and expand as SCRM approaches a more mature state.
The Open Group Trusted Technology Forum (OTTF) Protecting the Technology Supply Chain

Risks – Focusing on Tainted and Counterfeit Products

...pose a threat to the end-to-end product manufacturing / development process
O-TTPS Snapshot – Mitigating Risks for Tainted and Counterfeit Products

- **A tainted product** is “produced by the provider and is acquired through reputable channels but has been tampered with maliciously”. - Could result in:
  - product failure, degraded performance, weakened security mechanisms allowing rogue functionality and potentially critical damage

- **A counterfeit product** is “produced other than by or for the provider, or is supplied by other than a reputable channel, and is represented as legitimate”. – Could result in:
  - For customers: if product fails at critical juncture – loss of productivity, revenue
  - For providers: loss of revenue stream and brand damage

* NOTE: Malware & exploitable weaknesses could be introduced via poor manufacturing controls/practices (but not according to this definition); -- Malicious intent difficult to prove
# Technology Supply Chain Threat Matrix

<table>
<thead>
<tr>
<th>Taint</th>
<th>Counterfeit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Upstream</strong></td>
<td><strong>Provider</strong></td>
</tr>
<tr>
<td>Malware</td>
<td>✔️</td>
</tr>
<tr>
<td>Malicious code (masquerading as vulnerabilities)</td>
<td>✔️</td>
</tr>
<tr>
<td>Unauthorized “Parts”</td>
<td>✔️</td>
</tr>
<tr>
<td>Unauthorized Configuration</td>
<td></td>
</tr>
<tr>
<td>Scrap/Substandard Parts</td>
<td></td>
</tr>
<tr>
<td>Unauthorized Production</td>
<td></td>
</tr>
</tbody>
</table>
O-TTPS Snapshot

- The Standard/Snapshot – released March 9, 2012 – a set of prescriptive requirements and recommendations for organizational best practices
- Apply across product life cycle. Some highly correlated to threats of taint and counterfeit - others more foundational but considered essential.

- 2 areas of requirements – that often overlap depending on product and provider:
  - Technology Development mostly under the provider’s in-house supervision
  - Supply Chain activities mostly where provider interacts with third parties who contribute their piece in the product’s life cycle
O-TTPS: Technology Development Activities

- Product Development/Engineering Requirements in:
  - Software/Firmware/Hardware Design Process
  - Development/Engineering Process and Practices
  - Configuration Management
  - Quality/Test Management
  - Product Sustainment Management

- Secure Development/Engineering Requirements in:
  - Threat Analysis and Mitigation
  - Run-time Protection Techniques
  - Vulnerability Analysis and Response
  - Product Patching and Remediation
  - Secure Engineering Practices
  - Monitor and assess the impact of changes in the threat landscape.
O-TTPS: Supply Chain Activities

- Supply Chain Requirements In:
  - Risk Management
  - Physical Security
  - Access Controls
  - Employee and Supplier Security
  - Business Partner Security
  - Supply Chain Security Training
  - Information Systems Security
  - Trusted Technology Components
  - Secure Transmission and Handling
  - Open Source Handling
  - Counterfeit Mitigation
  - Malware Detection
Customer/Acquirer

Demands certificate as evidence of conformance to standards

Integrator

Will seek business partners who can meet Trusted Technology Provider requirements

Standards Body

Will seek ways of achieving market up-take/integrity of standards

Component Suppliers

May be hardware, software, global, open source - or not - multiple supplier layers

Provider

Will seek business partners who can meet Trusted Technology Provider requirements

Business Partners

Certification/Accreditation Body

Must be independent & vendor/technology-neutral

Business Partners
"Software Assurance in Acquisition: Mitigating Risks to the Enterprise"
Version 1.0, Oct 2008, available for community use;

1. Insert and enforce software assurance requirements in contracts.
2. Determine how much risk the organization can afford and who is accountable for that risk.
Executive Summary

1. Introduction
   1.1 Background
   1.2 Purpose and Scope
   1.3 Audience—Acquisition Official Defined
   1.4 Document Structure
   1.5 Risk-Managed Software Acquisition Process

2. Planning Phase
   2.1 Needs Determination, Risk Categorization, & Solution Alternatives
   2.2 SwA Requirements
   2.3 Acquisition Plan and/or Acquisition Strategy
   2.4 Evaluation Plan and Criteria
   2.5 SwA Due Diligence Questionnaires

3. Contracting Phase
   3.1 Request for Proposals
      3.1.1 Work Statement
      3.1.2 Terms and Conditions
      3.1.3 Instructions to Suppliers
      3.1.4 Certifications
      3.1.5 Prequalification
   3.2 Proposal Evaluation
   3.3 Contract Negotiation
   3.4 Contract Award

4. Implementation and Acceptance Phase
   4.1 Contract Work Schedule
   4.2 Change Control
   4.3 Risk Management Plan
   4.4 Assurance Case Management
   4.5 Independent Software Testing
   4.6 Software Acceptance

5. Follow-on Phase
   5.1 Support and Maintenance
      5.1.1 Risk Management
      5.1.2 Assurance Case Management—Transition to Ops
      5.1.3 Other Change Management Considerations
   5.2 Disposal or Decommissioning

Appendix A/B—Acronyms/Glossary
Appendix C—An Imperative for SwA in Acquisition
Appendix D—Software Due Diligence Questionnaires
   Table D-1. COTS Proprietary Software Questionnaire
   Table D-2. COTS Open-Source Software Questionnaire
   Table D-3. Custom Software Questionnaire
   Table D-4. GOTS Software Questionnaire
   Table D-5. Software Services
Appendix E—Other Examples of Due Diligence Questionnaires
Appendix F—Sample Language for the RFP and/or Contract
   F.1 Security Controls and Standards
   F.2 Securely Configuring Commercial Software
   F.3 Acceptance Criteria
   F.4 Certifications
   F.5 Sample Instructions to Offerors Sections
   F.6 Sample Work Statement Sections
   F.7 Open Web Application Security Project
   F.8 Certification of Originality

Appendix H—References
<table>
<thead>
<tr>
<th>SwA Concern Categories</th>
<th>Risks</th>
<th>Purpose for Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software History and Licensing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Process Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Security Training and Awareness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planning and Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Built-in Software Defenses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Assembly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Manufacture and Packaging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assurance Claims and Evidence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Change Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeliness of Vulnerability Mitigation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Individual Malicious Behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security “Track Record”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial History and Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organizational History</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foreign Interests and Influences</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Confidentiality Policies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Environment for Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Services and Monitoring</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Question</td>
<td>COTS Proprietary</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>11</td>
<td>Does the license/contract restrict the licensee from discovering flaws or disclosing details about software defects or weaknesses with others (e.g., is there a “gag rule” or limits on sharing information about discovered flaws)?</td>
<td>✓</td>
</tr>
<tr>
<td>12</td>
<td>Does the license/contract restrict communications or limit the licensee in any potential communication with third-party advisors about provisions for support (e.g., is there a “gag rule” or limits placed on the licensee that affect ability to discuss contractual terms or breaches) regarding the licensed or contracted product or service?</td>
<td>✓</td>
</tr>
<tr>
<td>13</td>
<td>Does software have a positive reputation? Does software have a positive reputation relative to security? Are there reviews that recommend it?</td>
<td>✓</td>
</tr>
<tr>
<td>14</td>
<td>Is the level of security where the software was developed the same as where the software will operate?</td>
<td></td>
</tr>
</tbody>
</table>

**Development Process Management**

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>COTS Proprietary</th>
<th>COTS Open-Source</th>
<th>GOTS</th>
<th>Custom</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>What are the processes (e.g., ISO 9000, CMMI, etc.), methods, tools (e.g., IDEs, compilers), techniques, etc. used to produce and transform the software (brief summary response)?</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>16</td>
<td>What security measurement practices and data does the company use to assist product planning?</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>17</td>
<td>Is software assurance considered in all phases of development? Explain.</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>18</td>
<td>How is software risk managed? Are anticipated threats identified, assessed, and prioritized?</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
### Table 1 – SwA Concern Categories -- (with interests relevant to security and privacy)

<table>
<thead>
<tr>
<th>SwA Concern Categories</th>
<th>Risks</th>
<th>Purpose for Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service Confidentiality Policies</strong></td>
<td>Without policies to enforce client data confidentiality/privacy, acquirer’s data could be at risk without service supplier liability.</td>
<td>To determine the service provider’s confidentiality and privacy policies and ensure their enforcement.</td>
</tr>
</tbody>
</table>

### Table 3 - Questions for Hosted Applications

<table>
<thead>
<tr>
<th>No.</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What are the customer confidentiality policies? How are they enforced?</td>
</tr>
<tr>
<td>2</td>
<td>What are the customer privacy policies? How are they enforced?</td>
</tr>
<tr>
<td>3</td>
<td>What are the policies and procedures used to protect sensitive information from unauthorized access? How are the policies enforced?</td>
</tr>
<tr>
<td>4</td>
<td>What are the set of controls to ensure separation of data and security information between different customers that are physically located in the same data center? On the same host server?</td>
</tr>
<tr>
<td>5</td>
<td>Who configures and deploys the servers? Are the configuration procedures available for review, including documentation for all registry settings?</td>
</tr>
<tr>
<td>7</td>
<td>What are the data backup policies and procedures? How frequently are the backup procedures verified?</td>
</tr>
<tr>
<td>11</td>
<td>What are the agents or scripts executing on servers of hosted applications? Are there procedures for reviewing the security of these scripts or agents?</td>
</tr>
<tr>
<td>12</td>
<td>What are the procedures and policies used to approve, grant, monitor and revoke access to the servers? Are audit logs maintained?</td>
</tr>
<tr>
<td>13</td>
<td>What are the procedures and policies for handling and destroying sensitive data on electronic and printed media?</td>
</tr>
<tr>
<td>15</td>
<td>What are the procedures used to approve, grant, monitor, and revoke file permissions for production data and executable code?</td>
</tr>
</tbody>
</table>
Many SwA Resources Focus On Development
Architecture and Design Considerations for Secure Software – SwA Pocket Guide*

The IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK) defines the design phase as both “the process of defining the architecture, components, interfaces, and other characteristics of a system or component” and “the result of [that] process.” The software design phase:

- is the software engineering life cycle activity where software requirements are analyzed in order to produce a description of the software’s internal structure that will serve as the basis for its implementation.
- consists of the architectural design and detailed design activities that follow the software requirements analysis phase and precedes software implementation in the SDLC.

This pocket guide includes the following topics:

- Basic Concepts
- Design Principles for Secure Software
- Architecture and Threat Modeling
- Secure Design Patterns
  - Architectural-level Patterns
  - Design-level Patterns
- Secure Session Management
- Design and Architectural Considerations for Mobile Applications
- Formal Methods and Architectural Design
- Design Review and Verification
- Key Architecture and Design Practices for Mitigating Exploitable Software Weaknesses
- Questions to Ask Developers

*Download FREE SwA Pocket Guides at https://buildsecurityin.us-cert.gov/swa*
# Apply Key Principles & Practices

<table>
<thead>
<tr>
<th>General Principle</th>
<th>Key Practices</th>
<th>Principle Design Conformance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize the number of high-consequence targets</td>
<td>Principle of least privilege</td>
<td>Minimizes the number of actors in the system granted high levels of privilege, and the amount of time any actor holds onto its privileges.</td>
</tr>
<tr>
<td></td>
<td>Separation of privileges, duties, and roles</td>
<td>Ensures that no single entity (human or software) should have all the privileges required to modify, delete, or destroy the system, components and resources.</td>
</tr>
<tr>
<td></td>
<td>Separation of domains</td>
<td>This practice makes separation of roles and privileges easier to implement.</td>
</tr>
<tr>
<td>Don’t expose vulnerable or high-consequence components</td>
<td>Keep program data, executables, and configuration data separated</td>
<td>Reduces the likelihood that an attacker who gains access to program data will easily locate and gain access to program executables or control/configuration data.</td>
</tr>
<tr>
<td></td>
<td>Segregate trusted entities from untrusted entities</td>
<td>Reduces the exposure of the software’s high-consequence functions from its high-risk functions, which can be susceptible to attacks.</td>
</tr>
<tr>
<td></td>
<td>Minimize the number of entry and exit points</td>
<td>Reduces the attack surface.</td>
</tr>
<tr>
<td></td>
<td>Assume environment data is not trustworthy</td>
<td>Reduces the exposure of the software to potentially malicious execution environment components or attacker-intercepted and modified environment data.</td>
</tr>
<tr>
<td></td>
<td>Use only trusted interfaces to environment resources</td>
<td>This practice reduces the exposure of the data passed between the software and its environment.</td>
</tr>
<tr>
<td>Deny attackers the means to compromise</td>
<td>Simplify the design</td>
<td>This practice minimizes the number of attacker-exploitable vulnerabilities and weaknesses in the system.</td>
</tr>
<tr>
<td></td>
<td>Hold all actors accountable</td>
<td>This practice ensures that all attacker actions are observed and recorded, contributing to the ability to recognize and isolate/block the source of attack patterns.</td>
</tr>
<tr>
<td></td>
<td>Timing, synchronization, and sequencing should be simplified to avoid issues.</td>
<td>Modeling and documenting timing, synchronization, and sequencing issues will reduce the likelihood of race conditions, order dependencies, synchronization problems, and deadlocks.</td>
</tr>
<tr>
<td></td>
<td>Make secure states easy to enter and vulnerable states difficult to enter</td>
<td>This practice reduces the likelihood that the software will be allowed to inadvertently enter a vulnerable state.</td>
</tr>
<tr>
<td></td>
<td>Design for controllability</td>
<td>This practice makes it easier to detect attack paths, and disengage the software from its interactions with attackers. Caution should be taken when using this approach since it can open a whole range of new attack vectors.</td>
</tr>
<tr>
<td></td>
<td>Design for secure failure</td>
<td>Reduces the likelihood that a failure in the software will leave it vulnerable to attack.</td>
</tr>
</tbody>
</table>
Leverage Attack Patterns

**CAPEC** – Common Attack Pattern Enumeration and Classification: capec.mitre.org

**Attack Patterns:**
- Blueprint for creating a specific type of attack
- Abstracted common attack approaches from the set of known exploits
- Capture the attacker’s perspective to aid software developers, acquirers and operators in improving the assurance profile of their software

**Use Attack Patterns to:**
- Guide definition of appropriate policies
- Guide creation of appropriate security requirements (positive and negative)
- Provide context for architectural risk analysis
- Guide risk-driven secure code review
- Provide context for appropriate security testing
- Provide a bridge between secure development and secure operations

Software security testing is not the same as testing the correctness and adequacy of security functions implemented by software, which are most often verified through requirements-based testing that:

- cannot fully demonstrate that software is free from exploitable weaknesses / vulnerabilities.
- is not the best approach to determining how software will behave under anomalous and hostile conditions.

Software Security Test Techniques throughout SDLC

Penetration Testing can enhance pre-deployment test outcomes and identify post-release exploit points.

Secure Coding

- Preparing to Write Secure Code
- Secure Coding Principles
- Secure Coding Practices
- Secure Memory and Cache Management
- Secure Error and Exception Handling
- What to Avoid
- Questions to Ask Developers

Are any compiler warnings disabled in code being delivered?
Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses

• Identifies mission/business risks attributable to the respective weaknesses; identifies common attacks that exploit those weaknesses, and provides recommended practices for preventing the weaknesses.

• CWE focuses on stopping vulnerabilities at the source by educating designers, programmers, and QA/testers on how to eliminate all too-common mistakes before software is even shipped.

• CWE Top-N lists serve as tools for education, training and awareness to help 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.

• Software managers and CIOs can use the CWE list as a measuring stick of progress in their efforts to secure their software.

Understand Assurance - Related Process Capability Expectations

Look to Standards for Assurance Process Detail

Build or Refine and Execute Your Assurance Processes

Measure Your Results

Mission/Business Process
Understand Your Business Requirements for Assurance

Information System

Understand Assurance-Related Process Capability Expectations

Organization Support

Adapted from: Paul Croll, Computer Sciences Corporation, August 2007
Assurance Process Reference Model

**Development Organization**

- **DO 1** Establish the assurance resources to achieve key business objectives
- **DO 2** Establish the environment to sustain the assurance program within the organization

**Development Project**

- **DP 1** Identify and manage risks due to vulnerabilities throughout the product and system lifecycle
- **DP 2** Establish and maintain assurance support from the project
- **DP 3** Protect project and organizational assets

**Development Engineering**

- **DE 1** Establish assurance requirements
- **DE 2** Create IT solutions with integrated business objectives and assurance
- **DE 3** Verify and Validate an implementation for assurance

**Enterprise Assurance Support**

- **ES 1** Establish and maintain organizational culture where assurance is an integral part of achieving the mission
- **ES 2** Establish and maintain the ability to support continued delivery of assurance capabilities
- **ES 3** Monitor and improve enterprise support to IT assets

**Acquisition and Supplier Management**

- **AM 1** Select, manage, and use effective suppliers and third party applications based upon their assurance capabilities.

**Create to facilitate Communication Across An Organization’s Multi-Disciplinary Stakeholders**

Created by Michele Moss, BAH, SwA Processes & Practices

[[Visit the original resource for more information]](https://buildsecurityin.us-cert.gov/swa/proself_assm.html)
Practical Measurement Framework for Software Assurance and Information Security

Oct 2008→Feb 09→May 09→

The Center for Internet Security

The CIS Security Metrics 2009

February 9

Consensus Metric Definitions

Organizations struggle to make cost-effective security investments, decision intelligence, security professional tasks, widely accepted and unambiguous metrics for decision support. CIS established a consensus team of one hundred (100) industry experts to address this need. The result is a set of standard metrics and data definitions that can be used across organizations to collect and analyze data on security process performance and outcomes.

This document contains twenty-two (22) metric definitions for an (8) important business functions: Incident Management, Vulnerability Management, Patch Management, Application Security, Configuration Management and Financial Metrics. Additional consensus metrics are currently being defined for those and additional business functions.
Software Assurance Curriculum Project

- **Vol I: Master of Software Assurance Reference Curriculum**
  
  In Dec 2010 the IEEE Computer Society and the ACM recognized the Master of Software Assurance (MSwA) Reference Curriculum as a certified master’s degree program in SwA—the first curriculum to focus on assuring the functionality, dependability, and security of software and systems.

- **Vol II: SwA Undergraduate Course Outlines**
  
  see [www.sei.cmu.edu/library/abstracts/reports/10tr019.cfm](http://www.sei.cmu.edu/library/abstracts/reports/10tr019.cfm) to download the PDF version of the report CMU/SEI-2010-TR-019

- **Vol III: Master of SwA Course Syllabi**

- **Vol IV: Community College Education**

  - Report on “Integrating the MSwA Reference Curriculum into Model Curriculum and Guidelines for Graduate Degree Programs in Information Systems” provides reference and guidance material.

  - To facilitate implementation, the MSwA project team is offering assistance, free of charge, to educational institutions looking to launch an MSwA degree program.

  - For more information, go to [https://buildsecurityin.us-cert.gov/bsi/1165-BSI.html](https://buildsecurityin.us-cert.gov/bsi/1165-BSI.html).
Organizations that provide security engineering & risk-based analysis throughout the lifecycle will have more resilient software products / systems.

“Build Security In” throughout the lifecycle

- **Attack Modeling**
- **Secure S/W Requirements Engineering**
- **Secure Design Principles & Practices**
- **Secure Programming Practices**
- **Test / Validation of Security & Resilience**
- **Secure Distribution/Deployment**
- **Documentation for Secure Use & Configuration**

- **Abuse Cases**
- **Security Requirements**
- **Risk Analysis**
- **Design Review**
- **Risk-based Test Plans**
- **Code Review**
- **Static/Dynamic Analysis**
- **Risk Analysis**
- **Penetration Testing**
- **Security Ops & Vulnerability Mgt**

**Library Updates with Patches**

**Organizational Process Assets cover:** governance, policies, standards, training, tailoring guidelines

- Leverage Software Assurance resources (freely available) to incorporate in training & awareness
- Modify SDLC to incorporate security processes and tools (should be done in phases by practitioners to determine best integration points)
- Avoid drastic changes to existing development environment and allow for time to change culture and processes
- Make the business case and balance the benefits
- Retain upper management sponsorship and commitment to producing secure software.

* Adopted in part from “Software Assurance: Mitigating Supply Chain Risks” (DHS NCSD SwA); “What to Test from a Security Perspective for the QA Professional” (Cigital) and “Neutralizing the Threat: A Case Study in Enterprise-wide Application Security Deployments” (HP Fortify Software & Accenture Security Technology Consulting)
We are engaged with many parts of the Community for Software Assurance-related standardization.
ISO/IEC JTC1

- ISO/IEC 27036—IT Security Techniques—Supplier Relationships
  - Techniques between acquirer and supplier for supply chain risk management

  - Published in October 2010.
  - As published, the document includes language-independent summaries of nearly 70 classes of vulnerabilities.
  - The working group is already drafting the 2nd Edition of the report which will add information specific to individual programming languages.

  **Use recent versions of compilers and DO NOT disable compiler warning flags**

- SC7: ISO/IEC 15026, System and Software Assurance
  - Publication of the standard, by both ISO/IEC and IEEE, in spring 2011.
System and software assurance focuses on the management of risk and assurance of safety, security, and dependability within the context of system and software life cycle. Terms of Reference changed: ISO/IEC JTC1/SC7 WG7, previously “System and Software Integrity” SC7 WG9.
ISO/IEC/IEEE 15026 Assurance Case

Set of structured assurance claims, supported by evidence and reasoning (arguments), that demonstrates how assurance needs have been satisfied.

- Shows compliance with assurance objectives
- Provides an argument for the safety and security of the product or service.
- Built, collected, and maintained throughout the life cycle
- Derived from multiple sources

Sub-parts
- A high level summary
- Justification that product or service is acceptably safe, secure, or dependable
- Rationale for claiming a specified level of safety and security
- Conformance with relevant standards & regulatory requirements
- The configuration baseline
- Identified hazards and threats and residual risk of each hazard / threat
- Operational & support assumptions

Attributes
- Clear
- Consistent
- Complete
- Comprehensible
- Defensible
- Bounded
- Addresses all life cycle stages
NRC Regulatory Guidance on Cyber Security

  - Directly relates to current NRC guidance on cyber security in the supply chain and SDLC of an ICS regulated by the agency.
  - Section C.12.2 “Supply Chain Protection” control drill down to the vendor level with requirements accountability for the RG 5.71 control baseline (Appendices B&C).
  - Section C.12.3 “Trustworthiness” requires developers employ software quality and validation methods to minimize flawed or malformed software; requires all tools to undergo commercial certification process.
  - Section C.12.5 “Developer Security Testing”

Software Security Assurance: Not just a good idea

• Many people responsible for protecting most critical infrastructure facilities have felt comfortable about security of their systems.
  – Facilities rely on industrial control systems (ICS) -- custom-built suites of systems that control essential mechanical functions of power grids, processing plants, etc -- usually not connected to the Internet, also known as "air-gapped."
  – Many industry owners, operators and regulators believed that this security model provided an infallible, invulnerable barrier to malicious cyber attacks from criminals and advanced persistent threat (APT) adversaries.

• National Defense Authorization Act (NDAA) -- which included a focus on software security (in Section 932, Strategy on Computer Software Assurance) -- serves as first cybersecurity law of 2011 and requires the U.S. Dept of Defense to develop a strategy for ensuring the security of software applications.

• Software Security Assurance, a set of practices for ensuring proactive application security, is key to making applications compliant with this new law.

“How Stuxnet Demonstrates That Software Assurance Equals Mission Assurance:
The rules of the game have changed,” by Rob Roy, Federal CTO of Fortify, an HP Company
Program Protection Plan Outline and Guidance as “Expected Business Practice”

What’s in the DoD Policy Memo?

- “Every acquisition program shall submit a PPP for Milestone Decision Authority review and approval at Milestone A and shall update the PPP at each subsequent milestone and the Full-Rate Production decision.”

- Expected business practice, effective immediately, and reflected in upcoming DoDI 5000.02 and DAG updates

The PPP is the Single Focal Point for All Security Activities on the Program

http://www.acq.osd.mil/se/pg/index.html#PPP
### Software Assurance Methods

#### Development Process
Apply assurance activities to the procedures and structure imposed on software development.

#### Operational System
Implement countermeasures to the design and acquisition of end-item software products and their interfaces.

#### Development Environment
Apply assurance activities to the environment and tools for developing, testing, and integrating software code and interfaces.

---

**Table 5.3-5-5: Application of Software Assurance Countermeasures (sample)**

<table>
<thead>
<tr>
<th>Development Process</th>
<th>Static Analysis p/a</th>
<th>Design Inspect</th>
<th>Code Inspect p/a</th>
<th>CVE p/a</th>
<th>CAPEC p/a</th>
<th>CWE p/a</th>
<th>Pen Test</th>
<th>Test Coverage p/a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developmental CPI SW</td>
<td>100/80%</td>
<td>Two Levels</td>
<td>100/80</td>
<td>100/80</td>
<td>100/80</td>
<td>100/80</td>
<td>Yes</td>
<td>75/50%</td>
</tr>
<tr>
<td>Developmental Critical Function SW</td>
<td>100/80%</td>
<td>Two Levels</td>
<td>100/80</td>
<td>100/70</td>
<td>100/70</td>
<td>100/70</td>
<td>Yes</td>
<td>75/50%</td>
</tr>
<tr>
<td>Other Developmental SW</td>
<td>none</td>
<td>One level</td>
<td>100/65</td>
<td>10/0</td>
<td>10/0</td>
<td>10/0</td>
<td>No</td>
<td>50/25%</td>
</tr>
<tr>
<td>COTS CPI and Critical Function SW</td>
<td>Vendor SwA</td>
<td>Vendor SwA</td>
<td>Vendor SwA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
<td>UNK</td>
</tr>
<tr>
<td>COTS (other than CPI and Critical Function) and NDI SW</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No</td>
<td>UNK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operational System</th>
<th>Failover Multiple Supplier Redundancy</th>
<th>Fault Isolation</th>
<th>Least Privilege</th>
<th>System Element Isolation</th>
<th>Input checking / validation</th>
<th>SW load key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developmental CPI SW</td>
<td>30%</td>
<td>All</td>
<td>all</td>
<td>yes</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>Developmental Critical Function SW</td>
<td>50%</td>
<td>All</td>
<td>All</td>
<td>yes</td>
<td>All</td>
<td>all</td>
</tr>
<tr>
<td>Other Developmental SW</td>
<td>none</td>
<td>Partial</td>
<td>none</td>
<td>None</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>COTS (CPI and CF) and NDI SW</td>
<td>none</td>
<td>Partial</td>
<td>All</td>
<td>None</td>
<td>Wrappers/all</td>
<td>all</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development Environment</th>
<th>SW Product</th>
<th>Source</th>
<th>Release testing</th>
<th>Generated code inspection p/a</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Compiler</td>
<td>No</td>
<td>Yes</td>
<td>50/20</td>
<td></td>
</tr>
<tr>
<td>Runtime libraries</td>
<td>Yes</td>
<td>Yes</td>
<td>70/none</td>
<td></td>
</tr>
<tr>
<td>Automated test system</td>
<td>No</td>
<td>Yes</td>
<td>50/none</td>
<td></td>
</tr>
<tr>
<td>Configuration management system</td>
<td>No</td>
<td>Yes</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>No</td>
<td>Yes</td>
<td>50/none</td>
<td></td>
</tr>
</tbody>
</table>

**Development Environment Access**
Controlled access; Cleared personnel only

---

**Additional Guidance in PPP Outline and Guidance**
Software Assurance

Software Assurance (SwA) is the level of confidence that software functions as intended and is free from vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.*

*Derived From: CNSSI-4009

Automation

Languages, enumerations, registries, tools, and repositories throughout the Lifecycle

Including design, coding, testing, deployment, configuration and operation
Automation is *one piece* of the SwA puzzle.
Many DHS, DoD, and NIST sponsored efforts are key to changing how software-based systems are developed, deployed & operated securely. These are (or are becoming used in) international standards.
Making Security Measurable (MSM): You Are Here

Software Assurance  Enterprise Security Management  Threat Management

Design  Deploy  Build  Assess  Test  Deploy

CWE, CAPEC, CWSS, CWRAF  CPE, CCE, OVAL, OCIL, XCCDF, AssetId, ARF  CVE, CWE, CAPEC, MAEC, CybOX, IODEF

Vulnerabilities
Exploits
Attacks
Malware
## Cyber Threats Emerged Over Time

### 1980’s
- Password guessing
- Exploiting known vulnerabilities
- Burglaries
- Packet spoofing
- Automated probes/scans
- Network mgmt. diagnostics
- Sniffers
- Executable code attacks (against browsers)
- Automated widespread attacks
- GUI intruder tools
- Disabling audits
- Internet social engineering attacks
- Password cracking
- Hijacking sessions
- Back doors

### 1990’s
- Automated widespread attacks using NNTP to distribute attack
- Widespread attacks on DNS infrastructure
- Email propagation of malicious code
- Automated probes/scans
- “Stealth”/advanced scanning techniques
- Sophisticated command & control
- Diffuse spyware
- Anti-forensic techniques
- Increase in wide-scale Trojan horse distribution
- Windows-based remote controllable Trojans (Back Orifice)

### 2000’s
- DDoS attacks
- Binary encryption
- Increase in tailored worms
- Internet social engineering attacks
- WWW attacks
- Widespread denial-of-service attacks
- Techniques to analyze code for vulnerabilities without source code
- Distributed attack tools
- Sophisticated command & control
- Diffuse spyware
- Anti-forensic techniques
- Home users targeted
- Increase in wide-scale Trojan horse distribution
- Windows-based remote controllable Trojans (Back Orifice)

### 2010’s
- Attack Sophistication
- Cyber Threats Emerged Over Time
Solutions Also Emerged Over Time

1980’s
- “stealth”/advanced scanning techniques
- widespread attacks using NNTP to distribute attack
- widespread attacks on DNS infrastructure
- GUI intruder tools
- hijacking sessions
- disabling audits
- Internet social engineering tricks
- password cracking
- exploitable known vulnerabilities
- packet spoofing
- password guessing

1990’s
- automated probes/scans
- sniffers
- network mgmt. diagnostics
- executable code attacks (against browsers)
- automated widespread attacks
- backdoors
- Internet social engineering tricks
- hijacking sessions
- disabling audits
- password cracking

2000’s
- email propagation of malicious code
- DDoS attacks
- binary encryption
- increase in tailored worms
- sophisticated command & control
- anti-forensic techniques
- home users targeted
- distributed attack tools
- increase in wide-scale Trojan horse distribution
- Windows-based remote controllable Trojans (Back Orifice)
- techniques to analyze code for vulnerabilities without source code
- widespread denial-of-service attacks

2010’s
- attack sophistication
- distributed attack tools
- diffuse spyware
- Windows-based remote controllable Trojans (Back Orifice)
Architecting Security with Information Standards for COIs

- Asset Management
- Vulnerability Management
- Configuration Management
- Threat Management
- System Development
- System Certification
- Intrusion Detection
- Incident Management
- Change Management
- Trust Management
- Identity Management
- Central Reporting
Asset Management
Vulnerability Management
Configuration Management
Threat Management
System Development
System Certification
Intrusion Detection
Incident Management
Change Management
Trust Management
Identity Management
Central Reporting
# Cyber Ecosystem Standardization Efforts

<table>
<thead>
<tr>
<th>Question</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>What IT systems do I have in my enterprise?</td>
<td>CPE (Platforms)</td>
</tr>
<tr>
<td>What known vulnerabilities do I need to worry about?</td>
<td>CVE (Vulnerabilities)</td>
</tr>
<tr>
<td>What vulnerabilities do I need to worry about right now?</td>
<td>CVSS (Scoring System)</td>
</tr>
<tr>
<td>How can I configure my systems more securely?</td>
<td>CCE (Configurations)</td>
</tr>
<tr>
<td>How do I define a policy of secure configurations?</td>
<td>XCCDF (Configuration Checklists)</td>
</tr>
<tr>
<td>How can I be sure my systems conform to policy?</td>
<td>OVAL (Assessment Language)</td>
</tr>
<tr>
<td>How can I be sure the operation of my systems conforms to policy?</td>
<td>OCIL (Interactive Language)</td>
</tr>
<tr>
<td>What weaknesses in my software could be exploited?</td>
<td>CWE (Weaknesses)</td>
</tr>
<tr>
<td>What attacks can exploit which weaknesses?</td>
<td>CAPEC (Attack Patterns)</td>
</tr>
<tr>
<td>How can we recognize malware &amp; share that info?</td>
<td>MAEC (Malware Attributes)</td>
</tr>
<tr>
<td>What observable behavior might put my enterprise at risk?</td>
<td>CybOX (Cyber Observables)</td>
</tr>
<tr>
<td>What events should be logged, and how?</td>
<td>CEE (Events)</td>
</tr>
<tr>
<td>How can I aggregate assessment results?</td>
<td>ARF (Assessment Results)</td>
</tr>
<tr>
<td>Question</td>
<td>SCAP Components</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>What IT systems do I have in my enterprise?</td>
<td>CPE (Platforms)</td>
</tr>
<tr>
<td>What known vulnerabilities do I need to worry about?</td>
<td>CVE (Vulnerabilities)</td>
</tr>
<tr>
<td>What vulnerabilities do I need to worry about right now?</td>
<td>CVSS (Scoring System)</td>
</tr>
<tr>
<td>How can I configure my systems more securely?</td>
<td>CCE (Configurations)</td>
</tr>
<tr>
<td>How do I define a policy of secure configurations?</td>
<td>XCCDF (Configuration Checklists)</td>
</tr>
<tr>
<td>How can I be sure my systems conform to policy?</td>
<td>OVAL (Assessment Language)</td>
</tr>
<tr>
<td>How can I be sure the operation of my systems conforms to policy?</td>
<td>OCIL (Interactive Language)</td>
</tr>
<tr>
<td>What weaknesses in my software could be exploited?</td>
<td>CWE (Weaknesses)</td>
</tr>
<tr>
<td>What attacks can exploit which weaknesses?</td>
<td>CAPEC (Attack Patterns)</td>
</tr>
<tr>
<td>How can we recognize malware &amp; share that info?</td>
<td>MAEC (Malware Attributes)</td>
</tr>
<tr>
<td>What observable behavior might put my enterprise at risk?</td>
<td>CybOX (Cyber Observables)</td>
</tr>
<tr>
<td>What events should be logged, and how?</td>
<td>CEE (Events)</td>
</tr>
<tr>
<td>How can I aggregate assessment results?</td>
<td>ARF (Assessment Results)</td>
</tr>
</tbody>
</table>

Standardization Efforts leveraged by the Security Content Automation Protocol (SCAP)
Efforts focused on mitigating risks and enabling more robust continuous monitoring and faster incident response

<table>
<thead>
<tr>
<th>What IT systems do I have in my enterprise?</th>
<th>• CPE (Platforms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>What known vulnerabilities do I need to worry about?</td>
<td>• CVE (Vulnerabilities)</td>
</tr>
<tr>
<td>What vulnerabilities do I need to worry about right now?</td>
<td>• CVSS (Scoring System)</td>
</tr>
<tr>
<td>How can I configure my systems more securely?</td>
<td>• CCE (Configurations)</td>
</tr>
<tr>
<td>How do I define a policy of secure configurations?</td>
<td>• XCCDF (Configuration Checklists)</td>
</tr>
<tr>
<td>How can I be sure my systems conform to policy?</td>
<td>• OVAL (Assessment Language)</td>
</tr>
<tr>
<td>How can I be sure the operation of my systems conforms to</td>
<td>• OCIL (Interactive Language)</td>
</tr>
<tr>
<td>policy?</td>
<td></td>
</tr>
<tr>
<td>What weaknesses in my software could be exploited?</td>
<td>• CWE (Weaknesses)</td>
</tr>
<tr>
<td>What attacks can exploit which weaknesses?</td>
<td>• CAPEC (Attack Patterns)</td>
</tr>
<tr>
<td>How can we recognize malware &amp; share that info?</td>
<td>• MAEC (Malware Attributes)</td>
</tr>
<tr>
<td>What observable behavior might put my enterprise at risk?</td>
<td>• CybOX (Cyber Observables)</td>
</tr>
<tr>
<td>What events should be logged, and how?</td>
<td>• CEE (Events)</td>
</tr>
<tr>
<td>How can I aggregate assessment results?</td>
<td>• ARF (Assessment Results)</td>
</tr>
</tbody>
</table>

New FISMA reporting requirements
Leverage Common Weakness Enumeration (CWE) to mitigate risks to mission/business domains

CWE is a formal list of software weakness types created to:
• Serve as a common language for describing software security weaknesses in architecture, design, or code.
• Serve as a standard measuring stick for software security tools targeting these weaknesses.
• Provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

Some Common Types of Software Weaknesses:
- Buffer Overflows, Format Strings, Etc.
- Structure and Validity Problems
- Common Special Element Manipulations
- Channel and Path Errors
- Handler Errors
- User Interface Errors
- Pathname Traversal and Equivalence
- Errors
- Authentication Errors
- Resource Management Errors
- Insufficient Verification of Data
- Code Evaluation and Injection
- Randomness and Predictability
“Making Security Measureable”: measurablesecurity.mitre.org

Sponsored by DHS with MITRE as technical lead

Resources provided for voluntary adoption

Open, community efforts that are free to use

XML-based

Some important things to note
What is the context?

Where can automation help - today?

What problems are we trying to solve?

Where do we start?
There are many definitions of “weakness” -- *in this context*

**A (software) weakness** is a property of software/systems that, under the right conditions, may permit unintended/unauthorized behavior.

*Common Weakness Enumeration (CWE) http://cwe.mitre.org/

There are many definitions of “vulnerability” -- *in this context*:

**A (software) vulnerability** is a collection of one or more weaknesses that contain the right conditions to permit unauthorized parties to force the software to perform unintended behavior (a.k.a. “is exploitable”)

*Common Vulnerabilities and Exposures (CVE) http://cwe.mitre.org/

* Part of ITU-T Cyber Information Exchange (CYBEX) series 1500; co-sponsored by DHS NCSD
Prioritizing weaknesses to be mitigated

Lists are a good start but they are designed to be broadly applicable

We would like a way to specify priorities based on business/mission risk
Common Weakness Risk Analysis Framework (CWRAF)

How do I identify which of the 900+ CWE’s are most important for my specific business domain, technologies and environment?

Common Weakness Scoring System (CWSS)

How do I rank the CWE’s I care about according to my specific business domain, technologies and environment?

How do I identify and score weaknesses important to my organization?
Leveraging Vignettes in Cyber Security Standardization for Key ICT Applications in various Domains

<table>
<thead>
<tr>
<th>Technology Groups</th>
<th>Business/Mission Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Applications</td>
<td>e-Commerce</td>
</tr>
<tr>
<td></td>
<td>Banking &amp; Finance</td>
</tr>
<tr>
<td></td>
<td>Energy (i.e., SmartGrid, gas transmission)</td>
</tr>
<tr>
<td></td>
<td>Chemical</td>
</tr>
<tr>
<td></td>
<td>Manufacturing</td>
</tr>
<tr>
<td></td>
<td>Shipping &amp; Transportation (i.e., rail, freight, ships, airlines, aerospace, postal)</td>
</tr>
<tr>
<td></td>
<td>National Defense (i.e., weapon systems, Intel networks, defense industrial base)</td>
</tr>
<tr>
<td></td>
<td>Homeland Security (CBP, TSA, etc.)</td>
</tr>
<tr>
<td></td>
<td>Secret Service (NSA, etc.)</td>
</tr>
<tr>
<td></td>
<td>Government (other than Nat/Def &amp; HS)</td>
</tr>
<tr>
<td></td>
<td>Incident response (law enforcement, security services, etc.)</td>
</tr>
<tr>
<td></td>
<td>Public Health</td>
</tr>
<tr>
<td></td>
<td>Food &amp; Water</td>
</tr>
<tr>
<td></td>
<td>Telecommunications</td>
</tr>
<tr>
<td></td>
<td>Teleworking</td>
</tr>
<tr>
<td></td>
<td>e-Voting</td>
</tr>
</tbody>
</table>

Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups.
### CWSS Scoring Engine

**CWSS Score**

<table>
<thead>
<tr>
<th>Score</th>
<th>CWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>CWE-79</td>
</tr>
<tr>
<td>95</td>
<td>CWE-78</td>
</tr>
<tr>
<td>94</td>
<td>CWE-22</td>
</tr>
<tr>
<td>94</td>
<td>CWE-434</td>
</tr>
<tr>
<td>94</td>
<td>CWE-798</td>
</tr>
<tr>
<td>93</td>
<td>CWE-120</td>
</tr>
<tr>
<td>93</td>
<td>CWE-250</td>
</tr>
<tr>
<td>92</td>
<td>CWE-770</td>
</tr>
<tr>
<td>91</td>
<td>CWE-829</td>
</tr>
<tr>
<td>91</td>
<td>CWE-190</td>
</tr>
<tr>
<td>91</td>
<td>CWE-494</td>
</tr>
<tr>
<td>90</td>
<td>CWE-134</td>
</tr>
<tr>
<td>90</td>
<td>CWE-772</td>
</tr>
<tr>
<td>90</td>
<td>CWE-476</td>
</tr>
<tr>
<td>90</td>
<td>CWE-131</td>
</tr>
</tbody>
</table>

*User-defined cutoff*

- CWSS Score Engine
- Most Important Weaknesses
- CWRAF/CWSS in a Nutshell

**“Vignette”**
Which static analysis tools find the CWE’s I care about?

CWE Coverage Claims Representation (CCR)

Set of CWE’s tool claims to cover

Tool A

Tool B

Tool C

Most Important Weaknesses (CWE’s)
CWRAF/CWSS Provides Risk Prioritization for CWE throughout Software Life Cycle

• Enables education and training to provide specific practices for eliminating software fault patterns;
• Enables developers to mitigate top risks attributable to exploitable software;
• Enables testing organizations to use suite of test tools & methods (with CWE Coverage Claims Representation) that cover applicable concerns;
• Enables users and operation organizations to deploy and use software that is more resilient and secure;
• Enables procurement organizations to specify software security expectations through acquisition of software, hosted applications and services.
When should I focus on Weaknesses and Vulnerabilities?

Focus on Weaknesses

A type of defect that *may be* exploitable.

Keep Weaknesses from becoming vulnerabilities

Focus on Vulnerabilities

Something in code that *can actually* be exploited.
Putting it all Together

What weaknesses are most important?

- CWE
- CWSS

What types of attacks exploit those weaknesses?

Does the system contain any of those weaknesses?

Does my testing cover all of those weaknesses?
Evolution of Standardized Representations - Sharing

- Vulnerabilities
- Weaknesses
- Attack Patterns
- Malware Behavior
- Cyber Observables
- Threat Indicators

Imports & Extends:
- Object
- Defined Objects
- Actions
What is a cyber observable?

- a measurable event or stateful property in the cyber domain
  - Some measurable events: a registry key is created, a file is deleted, an http GET is received, …
  - Some stateful properties: MD5 hash of a file, value of a registry key, existence of a mutex, …

Cyber Observable eXpression (CybOX) is a standardized language for encoding and communicating information about cyber observables (http://cybox.mitre.org)
What is STIX™
Structured Threat Information eXpression

Language

Specify Capture Characterize Communicate

Cyber Threat Information

Community-driven

Consistency Clarity Support automation
Structuring Threat Information for Sharing

Why were they doing it?

Why should you care about it?

What exactly were they doing?

What you are looking for

Who was doing it?

What were they looking to exploit?

Where was it seen?

What should you do about it?
STIX Architecture

Structured Threat Information eXpression (STIX)

Architecture v0.3

Who was doing it?

What exactly were they doing?

Why were they doing it?

Why should you care about it?

What you are looking for

Where was it seen?

What should you do about it?

What were they looking to exploit?
• Org C must understand each format in use and try to map across formats – sacrificing time and potentially losing information
• Duplication of effort at each organization in the exchange is expensive and does not scale
Enabling Cross-Vendor Sharing

- Org C only needs to understand one format – no need to map and no information loss
- Each vendor maps their internal representations to the common format once – efficient and scalable
## CybOX Tool Roadmap

<table>
<thead>
<tr>
<th>Tool</th>
<th>Class</th>
<th>Language</th>
<th>Availability</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenIOC -&gt; CybOX</td>
<td>Translator</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>CybOX -&gt; OpenIOC</td>
<td>Translator</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>Snort -&gt; CybOX</td>
<td>Translator</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>Email -&gt; CybOX</td>
<td>Transform</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>CybOX -&gt; Snort</td>
<td>Translator</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>IODEF -&gt; CybOX</td>
<td>Translator</td>
<td>Python</td>
<td>12/12</td>
<td>New BSD</td>
</tr>
<tr>
<td>CybOX -&gt; Suricata</td>
<td>Translator</td>
<td>Python</td>
<td>12/12</td>
<td>New BSD</td>
</tr>
<tr>
<td>Suricata -&gt; CybOX</td>
<td>Translator</td>
<td>Python</td>
<td>12/12</td>
<td>New BSD</td>
</tr>
<tr>
<td>CybOX Python Bindings</td>
<td>Bindings</td>
<td>Python</td>
<td>Now</td>
<td>New BSD</td>
</tr>
</tbody>
</table>

BSD license is a very open license, allowing you to do practically anything with the software. It’s less restrictive than the GPL, but more restrictive than the Public Domain. There are only a couple precepts that must be adhered to, when using this license:

- You are free to redistribute the software, in binary or source form, as long as the copyright, conditions and disclaimer are present.
- You cannot use the name of originating organization or contributors to promote derivatives of the software, without written consent.

If adhered to, you are free to modify, copy and redistribute BSD-licensed software in either source or binary form as you see fit. You are not required to return code or patches to the upstream BSD-licensed software. You are free to change the license, or charge for derivatives, of the software, be it commercial or proprietary.
# MAEC Tool Roadmap

<table>
<thead>
<tr>
<th>Tool</th>
<th>Class</th>
<th>Language</th>
<th>Current Support</th>
<th>Avail.</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAEC/CybOX Python</td>
<td>Bindings</td>
<td>Python</td>
<td>MAEC v2.1/CybOX 1.0</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>Bindings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAEC -&gt; OVAL</td>
<td>Translator</td>
<td>Python</td>
<td>MAEC v2.1</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>Anubis -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>MAEC v2.1/CybOX 1.0</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>ThreatExpert -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>MAEC v1.1</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>GFI Sandbox -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>MAEC v2.1</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>FireEye -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>n/a -- in develop.</td>
<td>TBD</td>
<td>New BSD</td>
</tr>
<tr>
<td>Norman Sandbox -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>n/a -- in develop.</td>
<td>TBD</td>
<td>New BSD</td>
</tr>
<tr>
<td>PEFile -&gt; MAEC</td>
<td>Translator</td>
<td>Python</td>
<td>n/a -- in develop.</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>MAEC -&gt; Suricata</td>
<td>Translator</td>
<td>Python</td>
<td>n/a – in develop.</td>
<td>12/12</td>
<td>New BSD</td>
</tr>
<tr>
<td>MAEC -&gt; HTML</td>
<td>Translator</td>
<td>XSLT</td>
<td>MAEC v2.1</td>
<td>Now</td>
<td>New BSD</td>
</tr>
<tr>
<td>MAEC Comparator</td>
<td>Analysis</td>
<td>Python</td>
<td>MAEC v2.1</td>
<td>Now</td>
<td>New BSD</td>
</tr>
</tbody>
</table>
automation can help...

**Construction**
- Common Weakness Enumeration (CWE)
- Common Attack Pattern Enumeration and Classification (CAPEC)
- CWE Coverage Claims Representation (CCR)

**Verification**
- Common Weakness Enumeration (CWE)
- Common Weakness Risk Analysis Framework (CWRAF)
- Common Weakness Scoring System (CWSS)
- Common Attack Pattern Enumeration and Classification (CAPEC)
- CWE Coverage Claims Representation (CCR)

**Deployment**
- Common Vulnerabilities and Exposures (CVE)
- Open Vulnerability Assessment Language (OVAL)
- Malware Attribute Enumeration and Characterization (MAEC)
- Cyber Observables eXpression (CybOX)
How do we prevent this next time?

Are we being attacked?

Who is attacking and what do they want?

Are we at risk?

Adapted from September 2010 SwA Forum, CERT RMM for Assurance, Lisa Young, SEI
SwA Forum – Next session: 5-7 Mar 2013 @ NIST in Gaithersburg, MD

SwA Working Group Sessions: June 2013 @ MITRE in McLean, VA

SwA Websites: [www.us-cert.gov/swa](http://www.us-cert.gov/swa)

Email: [software.assurance@dhs.gov](mailto:software.assurance@dhs.gov)

Making Security Measureable: [measurablesecurity.mitre.org](http://measurablesecurity.mitre.org)

See Language for sharing exchange of indicators and correlation of incident information -- Cyber Observables eXpression (CybOX) at [http://cybox.mitre.org](http://cybox.mitre.org)
IT/Software Supply Chain Management is a National Security & Economic Issue

- Adversaries can gain “intimate access” to target systems, especially in a global supply chain that offers limited transparency.

- Advances in science and technology will always outpace the ability of government and industry to react with new policies and standards:
  - National security policies must conform with international laws and agreements while preserving a nation’s rights and freedoms, and protecting a nation’s self interests and economic goals;
  - Forward-looking policies can adapt to the new world of global supply chains;
  - Standards for automation, processes, and products must mature to better address supply chain risk management, systems/software assurance, and the exchange of information and indicators for cyber security;
  - Assurance Rating Schemes for software products and organizations are needed.

- IT/software suppliers and buyers can take more deliberate actions to security-enhance their processes, practices and products to mitigate risks:
  - Government & Industry have significant leadership roles in solving this
  - Individuals can influence the way their organizations adopt security practices

Globalization will not be reversed; this is how we conduct business — To remain relevant, standards and capability benchmarking measures must address “assurance” mechanisms needed to manage IT/Software Supply Chain risks.
Next SwA Forum at 5-7 March 2013 at NIST in Gaithersburg MD

SOFTWARE ASSURANCE FORUM

“Building Security In”

https://buildsecurityin.us-cert.gov/swa

Joe Jarzombek, PMP, CSSLP
Director for Software Assurance
Cyber Security & Communications
Department of Homeland Security
Joe.Jarzombek@hq.dhs.gov
(703) 235-3673