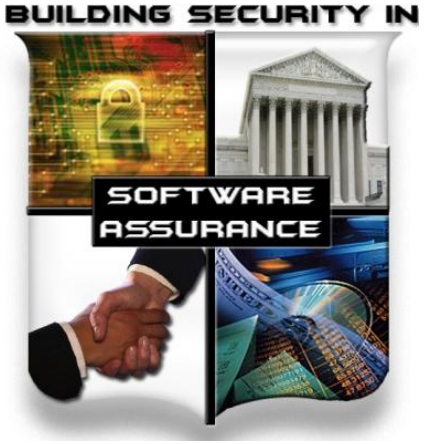
Software Assurance:

Measurable Security for FISMA, IAVA, and PPP Reporting Requirements and Criteria



Focusing on Program Protection Plans

August 1, 2012

Homeland Security

Joe Jarzombek, PMP, CSSLP
Director for Software Assurance
National Cyber Security Division

Measurable Security – FISMA, IAVA, and PPPs Reporting Requirements and Criteria

- Driven by challenges with software putting several missions at risk from a security perspective and last year's NDAA Section 932 "Strategy on Computer Software Assurance" the DoD now seems to be serious about mitigating exploitable software before it is used as an attack vector to breach military enterprises or compromise a weapon system.
- Measurement is needed to provide software assurance: "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."

What Are We Protecting?

Program Protection Planning

DODI 5000.02 Update

Technology

What: Leading-edge research and technology

Who Identifies: Technologists, System Engineers

ID Process: CPI Identification

Threat Assessment: Foreign collection threat informed by Intelligence and Counterintelligence assessments

Countermeasures: AT, Classification, Export Controls, Security, Foreign Disclosure, and CI activities

Focus: "Keep secret stuff in" by protecting any form of technology

Components

What: Mission-critical elements and components

Who Identifies: System Engineers, Logisticians

ID Process: Criticality Analysis

Threat Assessment: DIA SCRM TAC

Countermeasures: SCRM, SSE, Anti-counterfeits, software assurance, Trusted Foundry, etc.

Focus: "Keep malicious stuff out" by protecting key mission components

Information

What: Information about applications, processes, capabilities and end-items

Who Identifies: All

ID Process: CPI identification, criticality analysis, and classification guidance

Threat Assessment: Foreign collection threat informed by Intelligence and Counterintelligence assessments

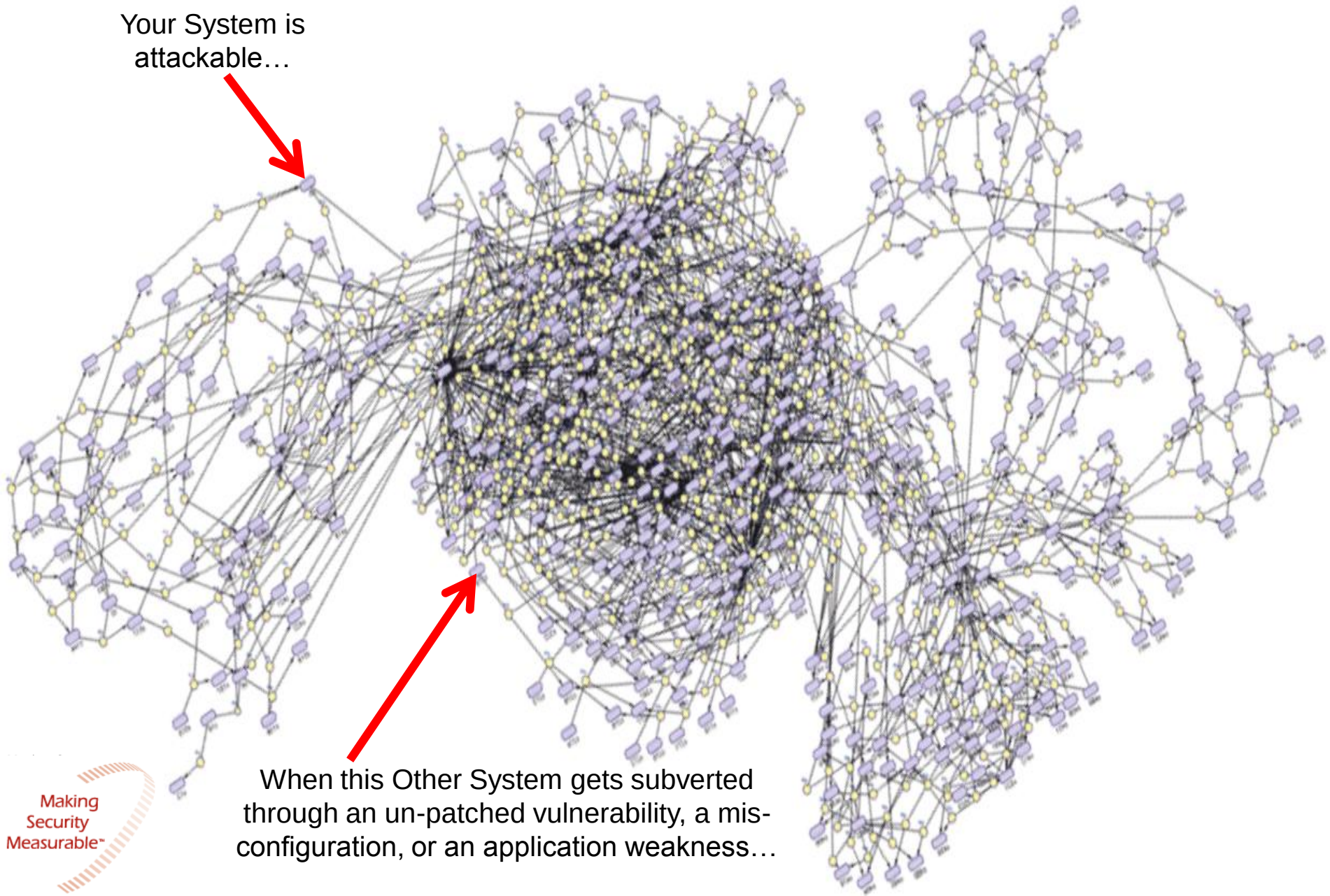
Countermeasures: Information Assurance, Classification, Export Controls, Security, etc.

Focus: "Keep critical information from getting out" by protecting data

Protecting Warfighting Capability Throughout the Lifecycle

Note: Program Protection Planning Includes DoDI 8500 series

Today Everything's Connected

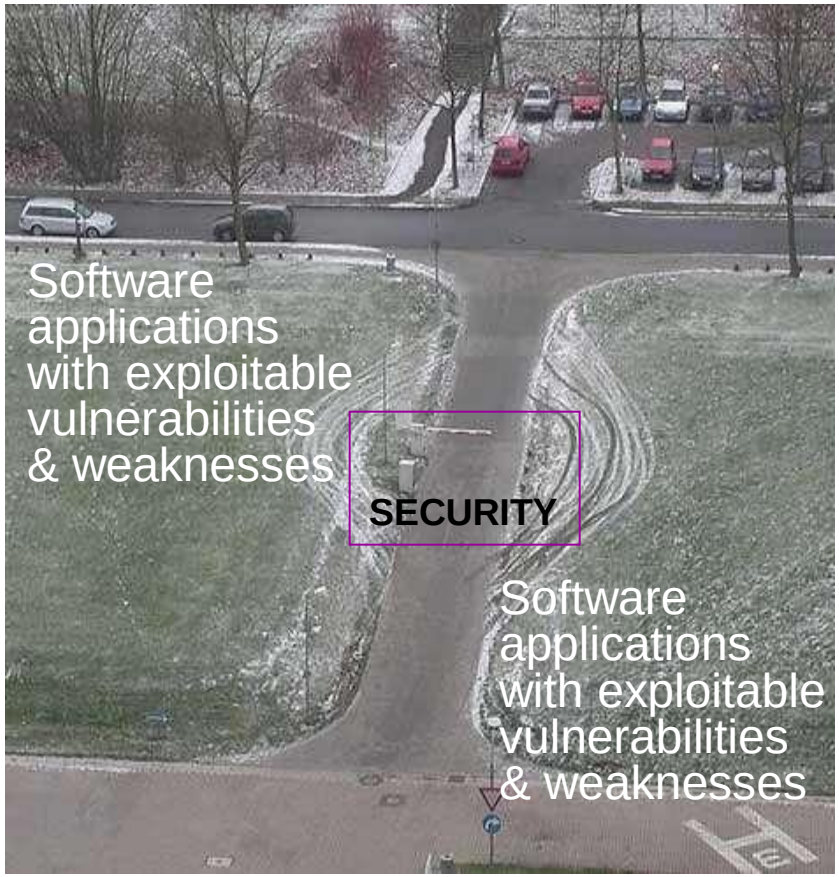


Making Security Measurable™

Exploitable Software Weaknesses are sources for future Zero-Day Attacks

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.

Homeland Security

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

Department of Defense (DoD) Software Assurance Definition

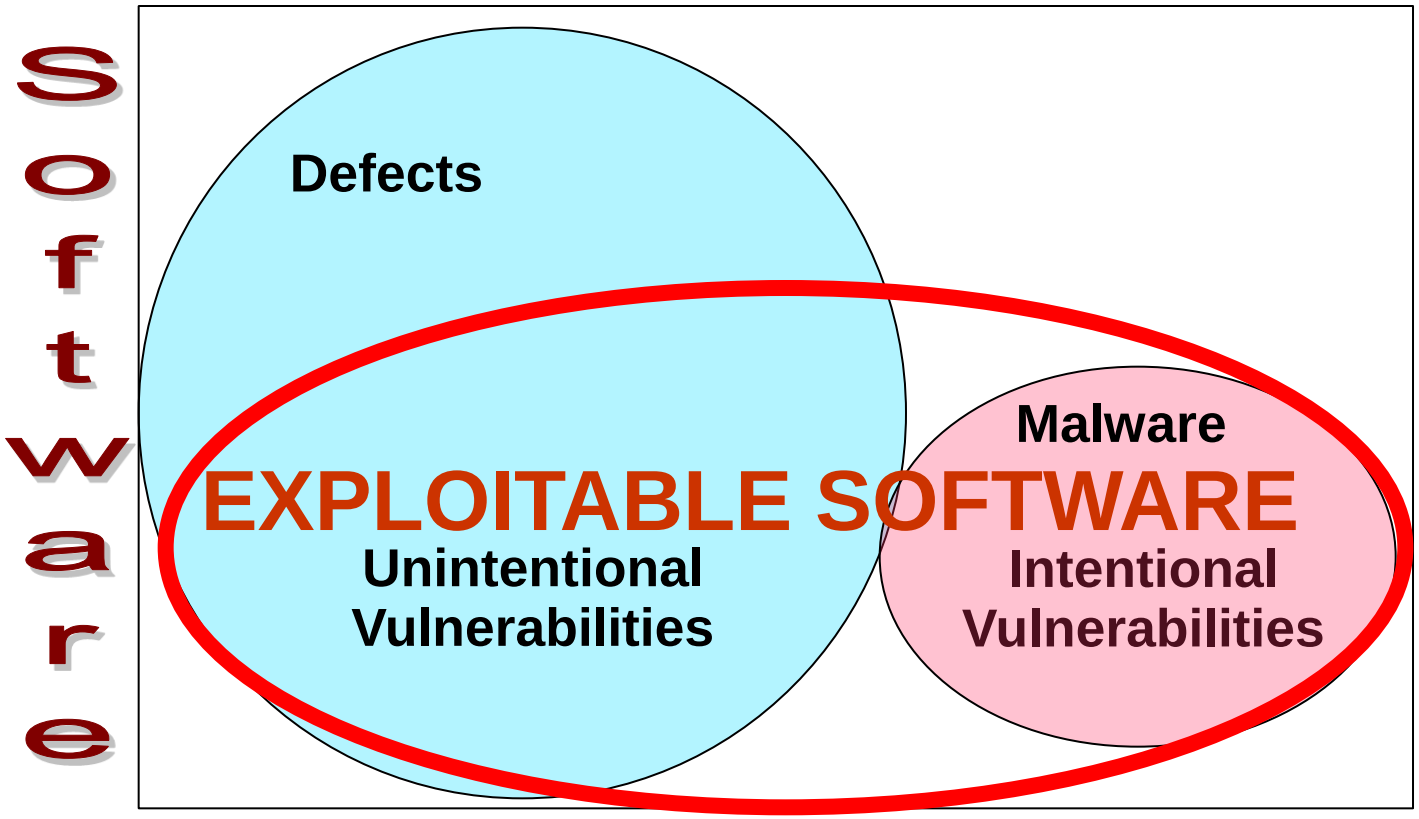
Software Assurance (SwA) is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.*

From CNSS Instruction 4009 (26APR2010), strengthened to address DoD's required roles and responsibilities across the Acquisition Life Cycle .

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".

Software Assurance Addresses Exploitable Software:
Outcomes of non-secure practices and/or malicious intent

Exploitation potential of vulnerability is independent of "intent"



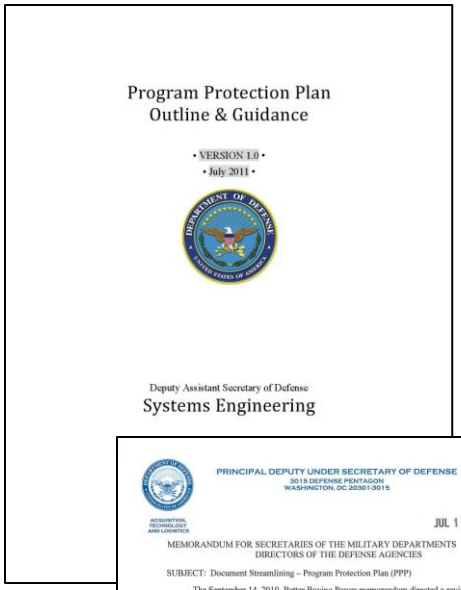
'High quality' can reduce security flaws attributable to defects; yet traditional S/W quality assurance does not address intentional malicious behavior in software

*Intentional vulnerabilities: spyware & malicious logic deliberately imbedded (might not be considered defects)

Homeland Security

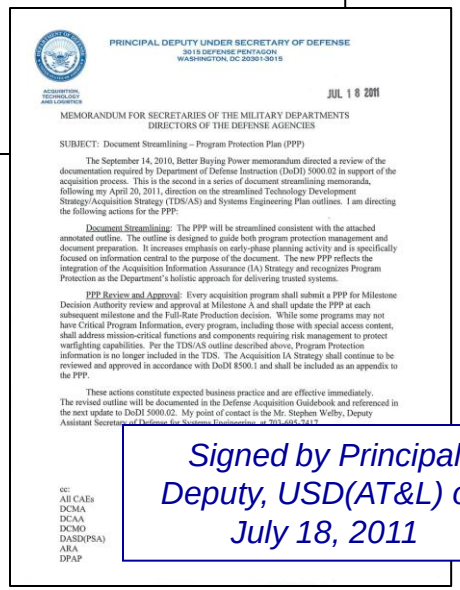
Note: Chart is not to scale – notional representation -- for discussions

Program Protection Plan Outline and Guidance as "Expected Business Practice"


Program Protection Plan
Outline & Guidance

• VERSION 1.0 •
• July 2011 •

Deputy Assistant Secretary of Defense
Systems Engineering


PRINCIPAL DEPUTY UNDER SECRETARY OF DEFENSE
3015 DEFENSE PENTAGON
WASHINGTON, DC 20301-3015

JUL 18 2011

MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS
DIRECTORS OF THE DEFENSE AGENCIES

SUBJECT: Document Streamlining – Program Protection Plan (PPP)

The September 14, 2010, Better Buying Power memorandum directed a review of the documentation required by Department of Defense Instruction (DoDI) 5000.02 in support of the acquisition process. This is the second in a series of document streamlining memoranda, following my April 20, 2011, direction on the streamlined Technology Development Strategy/Acquisition Strategy (TDS/AS) and Systems Engineering Plan outlines. I am directing the following actions for the PPP:

Document Streamlining: The PPP will be streamlined consistent with the attached annotated outline. The outline is designed to guide both program protection management and document preparation. It increases emphasis on early-phase planning activity and is specifically focused on information central to the purpose of the document. The new PPP reflects the integration of the Acquisition Information Assurance (IA) Strategy and recognizes Program Protection as the Department's holistic approach for delivering trusted systems.

PPP Review and Approval: 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. While some programs may not have Critical Program Information, every program, including those with special access content, shall address mission-critical functions and components requiring risk management to protect warfighting capabilities. Per the TDS/AS outline described above, Program Protection information is no longer included in the TDS. The Acquisition IA Strategy shall continue to be reviewed and approved in accordance with DoDI 8500.1 and shall be included as an appendix to the PPP.

These actions constitute expected business practice and are effective immediately. The revised outline will be documented in the Defense Acquisition Guidebook and referenced in the next update to DoDI 5000.02. My point of contact is the Mr. Stephen Welby, Deputy Assistant Secretary of Defense for Systems Engineering, at 703-692-7417.

cc:
All CAEs
DCMA
DCAA
DCMO
DASD(PSA)
ARA
DPAP

Signed by Principal Deputy, USD(AT&L) on July 18, 2011

What's in the 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

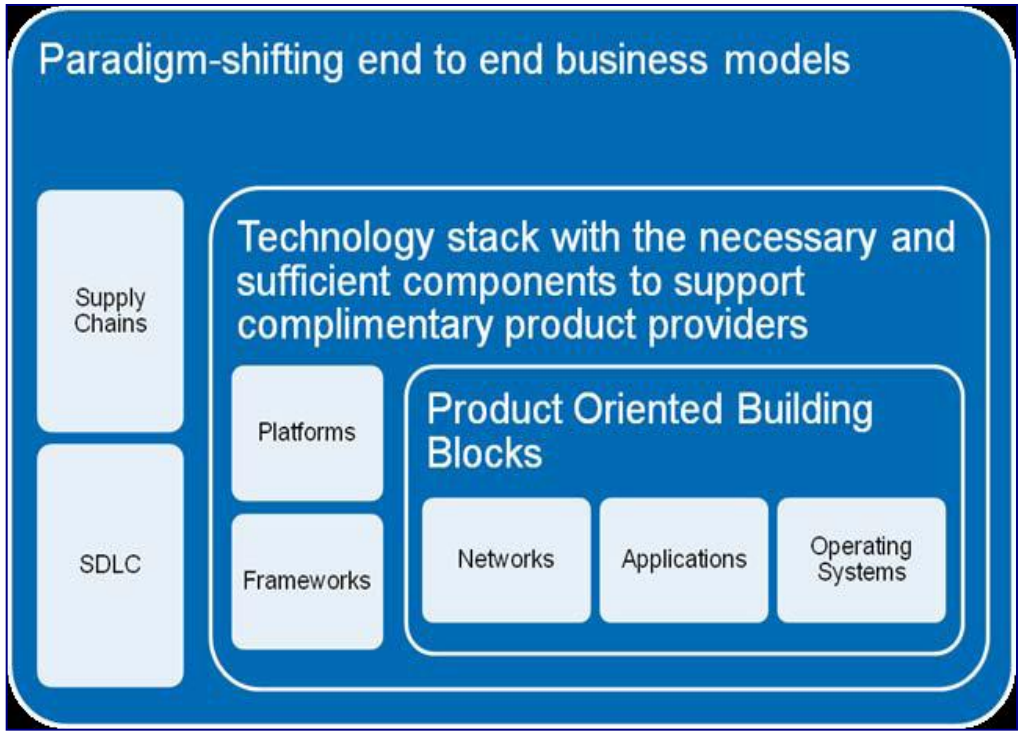
http://www.acq.osd.mil/se/pg/index.html#PPP

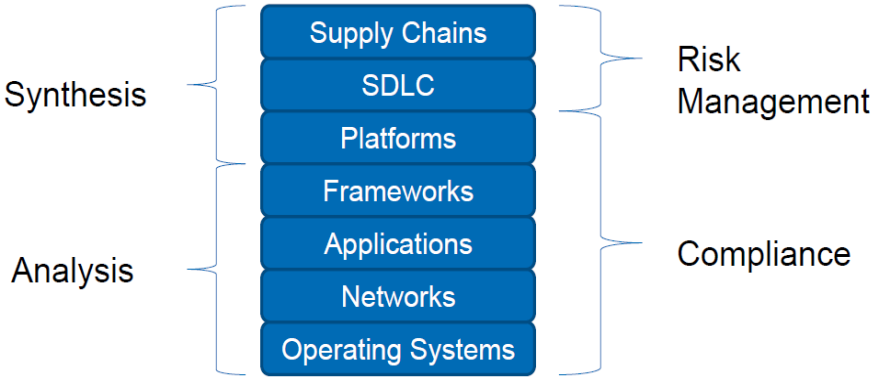
IT/software security risk landscape is a convergence between “defense in depth” and “defense in breadth”

Enterprise Risk Management and Governance are security motivators

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

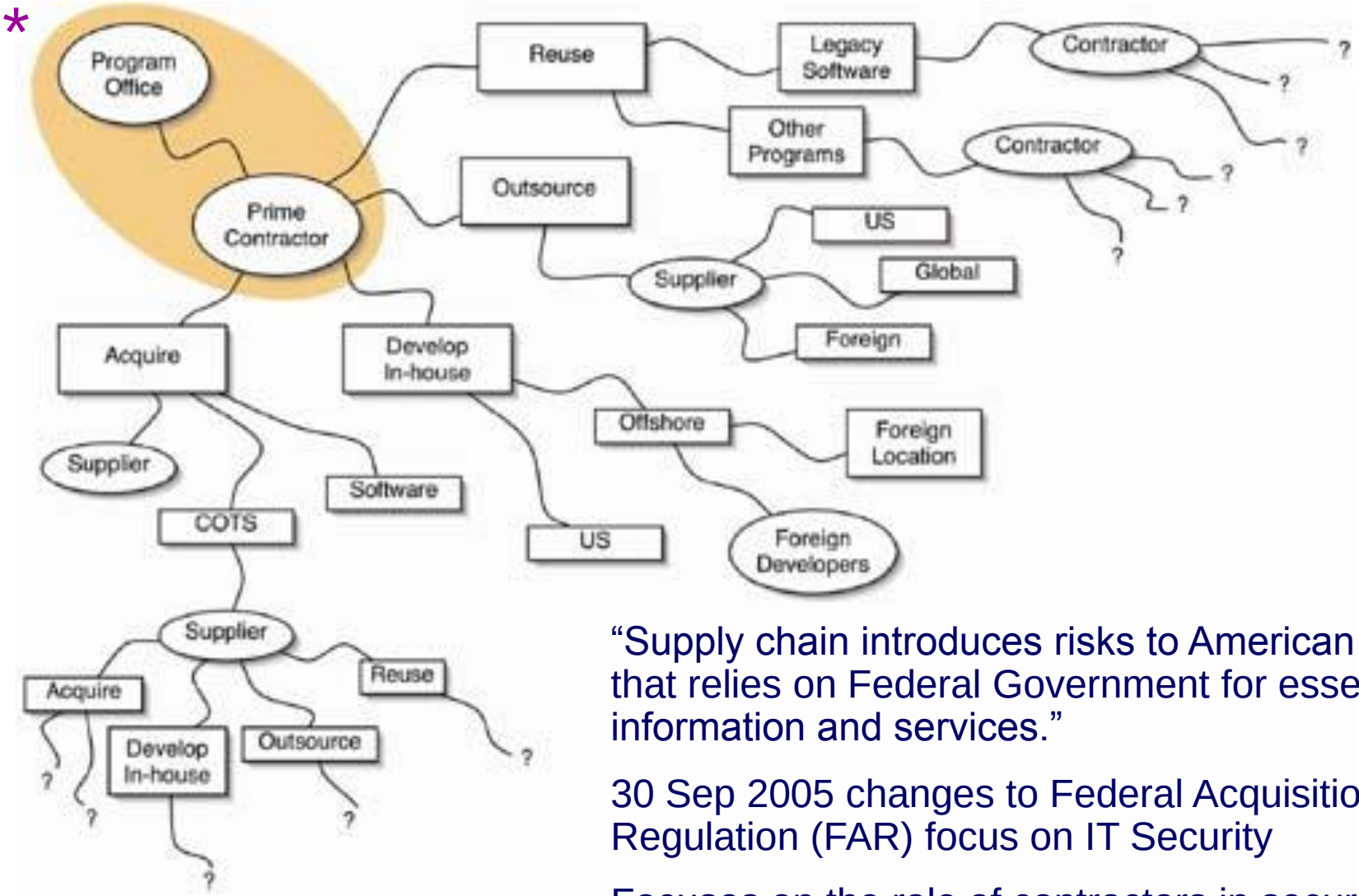
> “In the digital age, sovereignty is demarcated not by territorial frontiers but by supply chains.”
>
> – Dan Geer, CISO In-Q-Tel





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

*



"Supply chain introduces risks to American society that relies on Federal Government for essential information and services."

30 Sep 2005 changes to Federal Acquisition Regulation (FAR) focus on IT Security

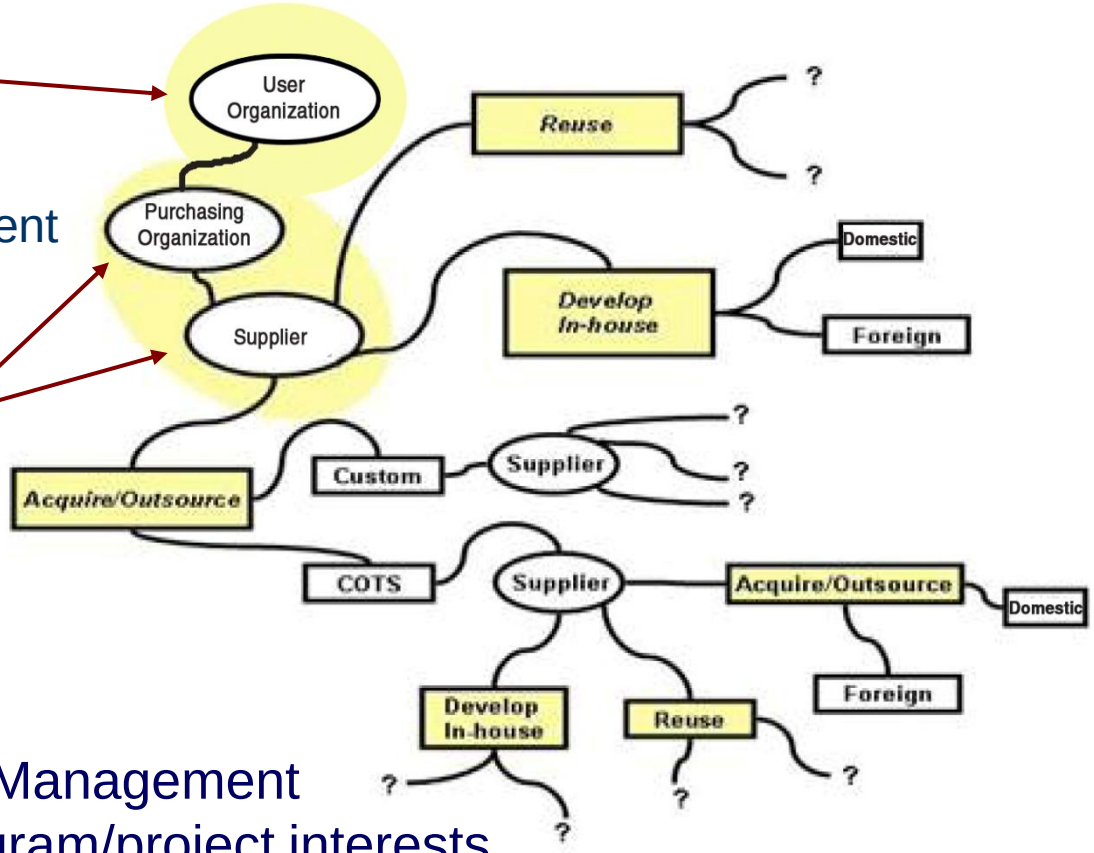
Focuses on the role of contractors in security as Federal agencies outsource various IT functions.

Homeland Security

"Scope of Supplier Expansion and Foreign Involvement" graphic in DACS www.softwaretechnews.com Secure Software Engineering, July 2005 article "Software Development Security: A Risk Management Perspective" synopsis of May 2004 GAO-04-678 report "Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks"

Risk Management (Enterprise ⟷ Project):
Shared Processes & Practices ⟷ Different Focuses

- Enterprise-Level:
  - Regulatory compliance
  - Changing threat environment
  - Business Case
- Program/Project-Level:
  - Cost
  - Schedule
  - Performance



Software Supply Chain Risk Management traverses enterprise and program/project interests

1. Insert and enforce software assurance requirements in contracts.
2. Review IT security policies to ensure that all users of organizational networks and data comply with the strictest security policies possible with respect to the mission.
3. Determine how much risk the organization can afford and who is accountable for that risk.

Homeland Security

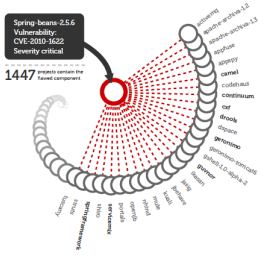
Thousands of downloads from open libraries with documented vulnerabilities

Security Vulnerabilities

It's not uncommon for vulnerabilities to be discovered in popular components. Updates that address the problem are typically provided quickly. However, even when security warnings are posted and easily accessible, they are often overlooked. In March 2009, the United States Computer Emergency Readiness Team and the National Institute of Standards and Technology (US-CERT/NIST) issued a warning that the Legion of the Bouncy Castle Java Cryptography API component was extremely vulnerable to remote attacks. In January 2011, almost 2 years later, 1,651 different organizations downloaded the vulnerable version of Bouncy Castle from the Central Repository within a single month. [v] In January 2010, the US-CERT/NIST posted an alert via their National Vulnerability Database that Jetty had a critical security flaw, which might allow attackers to execute arbitrary code, overwrite files and allow unauthorized disclosure of information. Regardless of the warning, in December of 2010, nearly a year later, approximately 11,000 different organizations downloaded the vulnerable version of Jetty from the Central Repository in a single month. [vi]

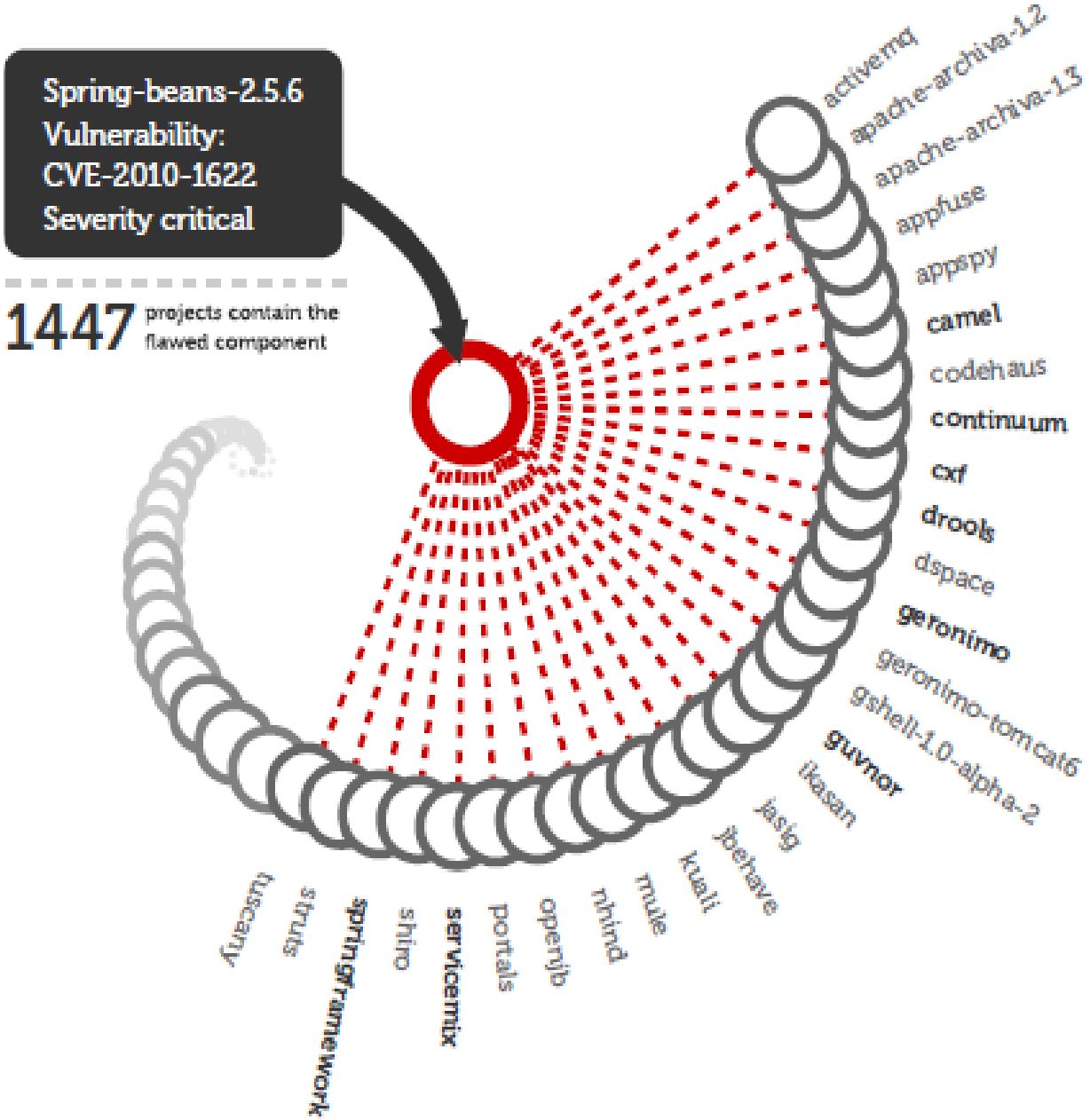
2 years after a vulnerability was discovered, organizations continue to download the flawed version of Bouncy Castle

Making the problem harder to deal with is the fact that a single vulnerability in a popular low level component may be used in hundreds, if not thousands of other commonly used open source projects as illustrated in Figure 6. You might not even be aware that your application uses the vulnerable component because it is buried multiple layers down in the open source component stack.



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

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

Challenges in Mitigating Risks Attributable to Exploitable Software and Supply Chains (cont.)

Enterprises seek comprehensive capabilities to:

- Avoid accepting software with **MALWARE** pre-installed. **MAEC**
- 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. **CVE**
- Determine that exploitable software **WEAKNESSES** that put the users most at risk are mitigated prior to operational acceptance or after put into use (and not previously evaluated for exploit potential). **CWE**

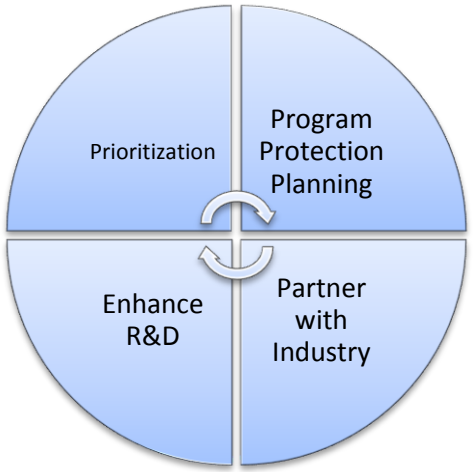
Homeland Security

International Community System Assurance Activities

- **ISO/IEC 15026 – System and Software Engineering – Systems and Software Assurance**
  - Establishes common assurance concepts, vocabulary, integrity levels and lifecycle
- **ISO/IEC 27036—IT Security Techniques—Supplier Relationships**
  - Establishes techniques between acquirer and supplier for supply chain risk management
- **International Council on Systems Engineering (INCOSE)**
  - Systems Security Engineering (SSE) working group established to develop SSE updates to INCOSE SE Handbook
- **The Open Group (TOG)**
  - The Open Trusted Technology Provider Framework (O-TTPF) - open standard that codifies best practices across the entire lifecycle covering:
    - Product Development
    - Secure Engineering
    - Supply Chain Integrity
    - http://www.opengroup.org/ogttf/

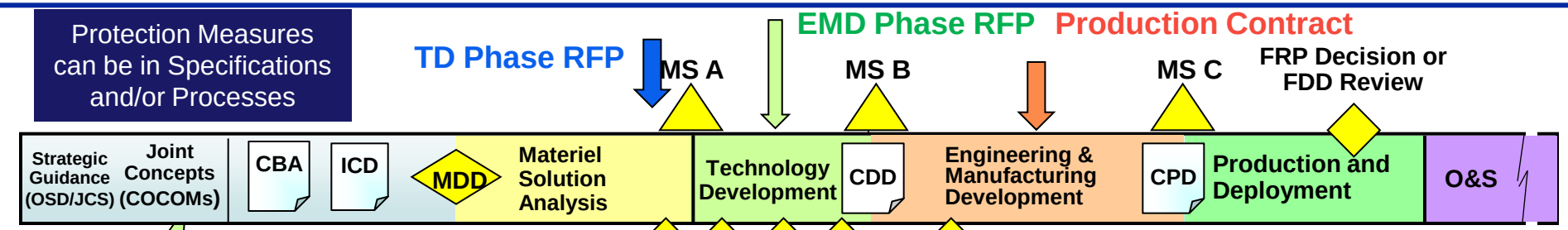
Trusted Defense Systems Strategy Basic Tenets

- **Prioritization:**
  - Focus security requirements on mission critical systems
  - Within systems, identify and protect critical components, technology, information
- **Comprehensive Program Protection Planning**
  - Early lifecycle identification of critical components
  - Provide PMs with analysis of supply chain risk
  - Protect critical components through trusted suppliers, or secure systems design
  - Assure systems through advanced vulnerability detection, test and evaluation
  - Manage counterfeit risk through sustainment
- **Partner with Industry**
  - Develop commercial standards for secure products
- **Enhance capability through R&D**
  - Leverage and enhance vulnerability detection tools and capabilities
  - Technology investment to advance secure software, hardware, and system design methods



Program Protection in Engineering Discipline

Protection Measures can be in Specifications and/or Processes



Generic RFP Language is Available

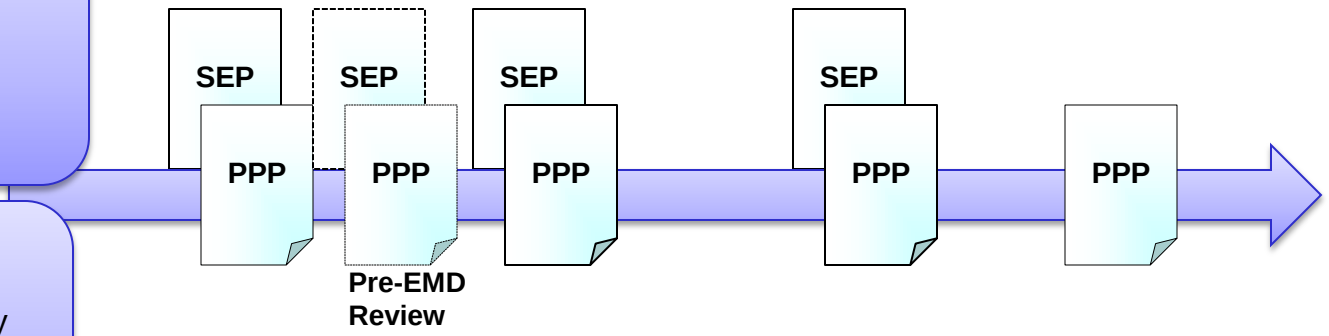
Focus Scope of Protection

Program Protection Analysis at SE Technical Reviews

Integrated Process to Manage Security Risks
- Foreign Collection
- Design Vulnerability
- Supply Chain Exploit/Insertion

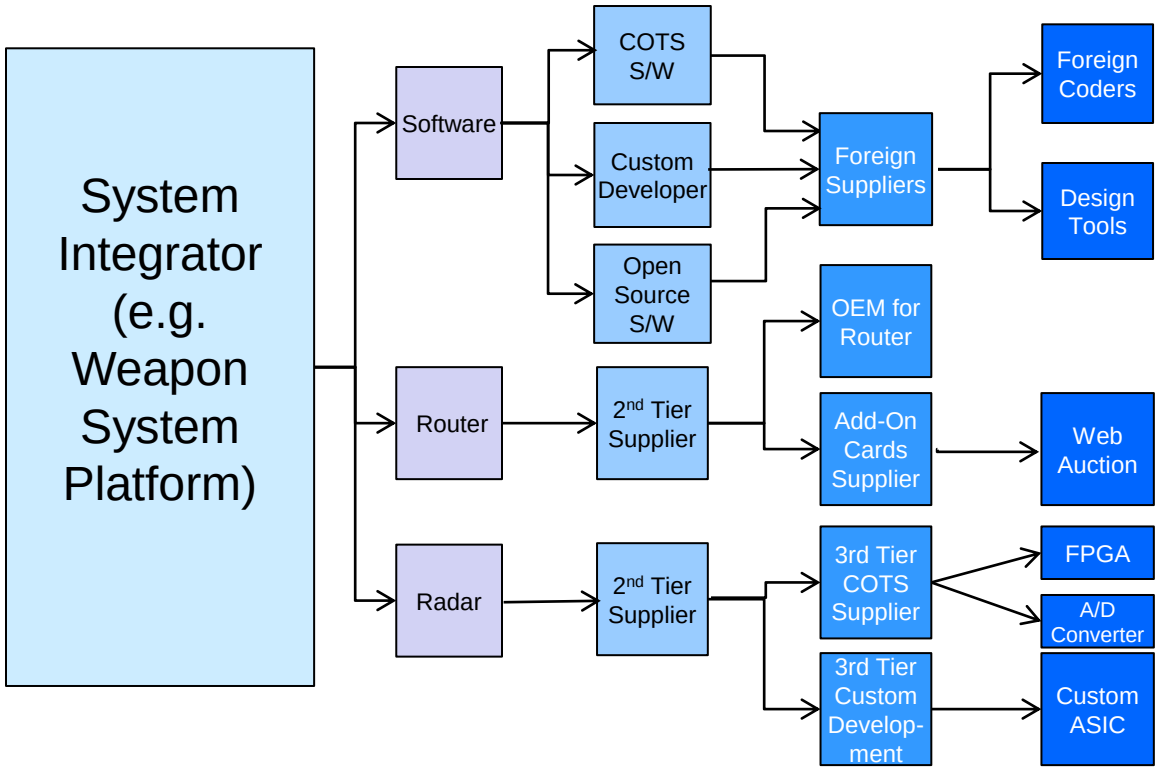
Protect Capability from Supply Chain/System Design Exploit
- Supply Chain Risk Management
- Software Assurance
- Information Assurance

Protect Advanced Technology Capability from Foreign Collection/Design Vulnerability
- Anti-Tamper
- Export Control
- Intel/CI/Security



Emphasizing Use of Affordable, Risk-based Countermeasures

Tiered Supply Chain Problem
(Notional Example)



Supplier Threat can reside several layers down from System Integrator

How is it shipped?
How is it verified and validated ?
How is it physically protected?
Do you execute a Blind Buy?
…

Manage Risks
Criticality
Schedule
Cost

1st Tier Supplier
2nd Tier Supplier
3rd Tier Supplier
4th Tier Supplier

PPP Methodology

Input Analysis Results:

Criticality Analysis Results

Mission	Critical Functions	Logic-Bearing Components (HW, SW, Firmware)	System Impact (I, II, III, IV)	Rationale
Mission 1	CF 1	Processor X	II	Redundancy
	CF 2	SW Module Y	I	Performance
Mission 2	CF 3	SW Algorithm A	II	Accuracy
	CF 4	FPGA 123	I	Performance

Supplier Risk Analysis Results

Supplier	Critical Components (HW, SW, Firmware)	Analysis Findings
Supplier 1	Processor X	Supplier Risk
	FPGA 123	Supplier Risk
Supplier 2	SW Algorithm A	Cleared Personnel
	SW Module Y	Cleared Personnel

Vulnerability Assessment Results

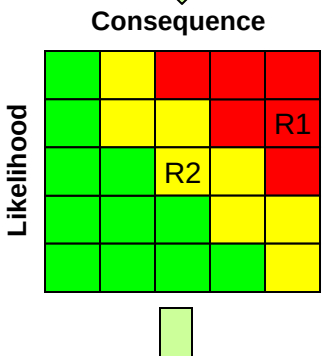
Critical Components (HW, SW, Firmware)	Identified Vulnerabilities	Exploit-ability	System Impact (I, II, III, IV)	Exposure
Processor X	Vulnerability 1 Vulnerability 4	Low Medium	II	Low Low
SW Module Y	Vulnerability 1 Vulnerability 2 Vulnerability 3 Vulnerability 6	High Low Medium High	I	High Low Medium Low
SW Algorithm A	None	Very Low	II	Very Low
FPGA 123	Vulnerability 1 Vulnerability 23	Low Low	I	High High

Risk Mitigation and Countermeasure Options

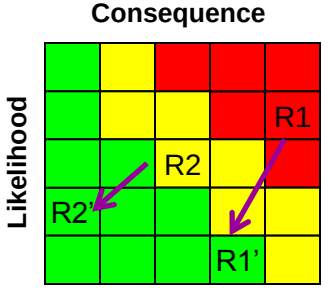
Consequence of Losing Mission Capability
Very High
High
Moderate
Low
Very Low

Likelihood of Losing Mission Capability
Near Certainty (VH)
Highly Likely (H)
Likely (M)
Low Likelihood (L)
Not Likely (VL)

Initial Risk Posture



Risk Mitigation Decisions



Criticality Analysis Methodology

Criticality Analysis

Inputs:

- ICD
- CDD
- Concept of Operations
- Concept of Employment
- Software development processes
- Sources and performance experience of key data handling components
- System architecture down to component level
- Vulnerabilities
- Verification plans
- WBS
- Etc.

Leverage existing mission assurance analysis, including flight & safety critical

Identify and Group Mission Threads by Priority

Identify Critical Functions Assign Criticality Levels

Map Threads and Functions to Subsystems and Components

Identify Critical Suppliers

Criticality Levels

Level I: **Total Mission Failure**

Level II: **Significant/Unacceptable Degradation**

Level III: **Partial/Acceptable Degradation**

Level IV: **Negligible**

Outputs:

- Table of Level I & II Critical Functions and Components
- TAC Requests for Information

Supplier Risk Analysis

Supplier Risk Analysis

- **Critical Logic-Bearing components** and **(Commercial) Software**
- **Contractor provides input data to the Program Office :**
  - Identify suppliers of logic-bearing devices and software/firmware modules
  - Provide *who* is designing, building, testing, and distributing critical components and *where*
- **Use intelligence community to conduct All-Source Counterintelligence Analyses**
  - Identifies foreign intelligence connections with suppliers of critical components
  - only provides threat analysis; doesn't provide solutions
  - Does not approve or qualify suppliers
  - Does not disqualify any suppliers

Vulnerability Assessment

Inputs:
System Architecture
Critical Functions and components
Concept of Operations
Software development processes
Procurement Processes (COTS)
Maintenance and sustainment processes
Supply Chain
CVE/CWE

Identify Potential Attack Vectors

Evaluate design and processes based on attack vectors

Assess exploitability of each attack vector

Determine overall exposure

Outputs:
- Table of Vulnerabilities to
  - critical functions and critical components
  - supply chain and development processes
- Potential countermeasures

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://cve.mitre.org/

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

Supply Chain Risk Mitigation Considerations (2/2)

- **Does the Developer Have:**
  - A design and code inspection process that requires specific secure design and coding standards as part of the inspection criteria?
  - Secure design and coding standards which considers CWE, Software Engineering Institute (SEI) *Top 10* secure coding practices and other sources when defining the standards?
- **Have Software Vulnerabilities Been Mitigated?**
  - Derived From Common Weakness Enumeration (CWE)
  - Common Vulnerabilities and Exposures (CVE)
  - Common Attack Pattern Enumeration and Classification (CAPEC)
- **Are Static Analysis Tools Used to Identify and Mitigate Vulnerabilities?**
- **Does the Software Contain Fault Detection/Fault Isolation (FDFI) and Tracking or Logging of Faults?**
- **Do the Software Interfaces Contain Input Checking and Validation?**
- **Is Access to the Development Environment Controlled With Limited Authorities and Does it Enable Tracing All Code Changes to Specific Individuals?**
- **Are Specific Code Test-Coverage Metrics Used to Ensure Adequate Testing?**
- **Are Regression Tests Routinely Run Following Changes to Code?**

Software Assurance Risk Mitigation Approaches for Security (1/3)

Counter-measure Selection

- **Development and integration environment for critical functions**
  - **Secure design and coding standards**
    - Authentication
    - Safe memory allocation and management
    - Input validation
    - Security-aware error and exception handling
    - Non-informative error messages
  - **Vulnerability and weakness data bases**
    - CVE to identify and coordinate SW vulnerabilities that allow attacker exploitation
    - CWE to examine software architecture/design and source code for weaknesses
    - CAPEC for the analysis of common destructive attack patterns
  - **Design and code inspections for security based upon secure design and code standards**
  - **Code scanning and correction**
    - Identify which vulnerabilities types will be detected and resolved as part of the contract
      - Support with static analysis tools or inspections
  - **Penetration testing**
  - **Alignment with SCRM (supply chain integrity)**
    - Ensure SW of known pedigree to counter malicious insertion for remote exploitation

Software Assurance Risk Mitigation Approaches for Security (2/3)

Counter-measure Selection

- **Operational system critical function design considerations**
  - **Fail over / multiple supplier redundancy**
  - **Fault isolation**
  - **Least privilege and separation of privilege**
  - **System element (critical Function) Isolation**
  - **SW Load Key**
  - **Alignment with SCRM (supply chain integrity)**
    - Ensure SW with secure interfaces and networks to counter cyber attacks
- **Development / Integration Environment Tools Security**
  - **Pedigree**
  - **Availability of source code for inspection / scanning**
  - **Release Inspection and testing for malicious insertion**
  - **Inspection of generated code**

Software Assurance Risk Mitigation Approaches for Security (3/3)

Counter-measure Selection

- **Alignment with FY11 NDAA Section 932 Software Assurance (SwA) Strategy**
  - Include contract requirements for SwA
  - Include SwA in milestone reviews and approvals
  - Include rigorous T&E of SwA in development, acceptance, and operational tests
    - Assure the security of software and software applications during software development
    - Detect vulnerabilities during testing of software
  - Detect intrusions during real-time monitoring of software applications
  - Remediation in legacy systems of critical SwA deficiencies

Software Assurance Methods

Counter-measure Selection

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)

Development Process								
Software (CPI, critical function components, other software)	**Static Analysis p/a**	**Design Inspect**	**Code Inspect p/a**	**CVE p/a**	**CAPEC p/a**	**CWE p/a**	**Pen Test**	**Test Coverage p/a**
Developmental CPI SW	100/80%	Two Levels	100/80	100/60	100/60	100/60	Yes	75/50%
Developmental Critical Function SW	100/80%	Two Levels	100/80	100/70	100/70	100/70	Yes	75/50%
Other Developmental SW	none	One level	100/65	10/0	10/0	10/0	No	50/25%
COTS CPI and Critical Function SW	Vendor SwA	Vendor SwA	Vendor SwA	0	0	0	Yes	UNK
COTS (other than CPI and Critical Function) and NDI SW	No	No	No	0	0	0	No	UNK

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

Development Environment								
SW Product	**Source**	**Release testing**	**Generated code inspection p/a**					
C Compiler	No	Yes	50/20					
Runtime libraries	Yes	Yes	70/none					
Automated test system	No	Yes	50/none					
Configuration management system	No	Yes	NA					
Database	No	Yes	50/none					
Development Environment Access	Controlled access; Cleared personnel only							

Additional Guidance in PPP Outline and Guidance

Risk Cost Benefit Trade-off

Mitigation

Input Analysis Results:

Criticality Analysis Results

Mission	Critical Functions	Logic-Bearing Components (HW, SW, Firmware)	System Impact (I, II, III, IV)	Rationale
Mission 1	CF 1	Processor X	II	Redundancy
	CF 2	SW Module Y	I	Performance
Mission 2	CF 3	SW Algorithm A	II	Accuracy
	CF 4	FPGA 123	I	Performance

Supplier Risk Analysis Results

Supplier	Critical Components (HW, SW, Firmware)	Analysis Findings
Supplier 1	Processor X	Supplier Risk
	FPGA 123	Supplier Risk
Supplier 2	SW Algorithm A	Cleared Personnel
	SW Module Y	Cleared Personnel

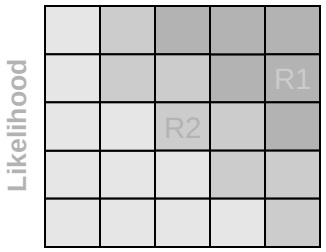
Vulnerability Assessment Results

Critical Components (HW, SW, Firmware)	Identified Vulnerabilities	Exploit-ability	System Impact (I, II, III, IV)	Exposure
Processor X	Vulnerability 1 Vulnerability 4	Low Medium	II	Low Low
SW Module Y	Vulnerability 1 Vulnerability 2 Vulnerability 3 Vulnerability 6	High Low Medium High	I	High Low Medium Low
SW Algorithm A	None	Very Low	II	Very Low
FPGA 123	Vulnerability 1 Vulnerability 23	Low Low	I	High High

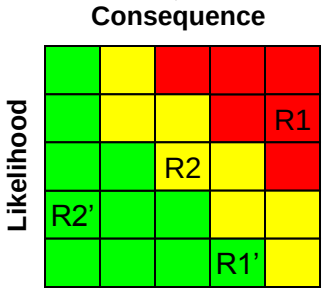
Consequence of Losing Mission Capability
Very High
High
Moderate
Low
Very Low

Likelihood of Losing Mission Capability
Near Certainty (VH)
Highly Likely (H)
Likely (M)
Low Likelihood (L)
Not Likely (VL)

Initial Risk Posture



Risk Mitigation Decisions



Documented trade-off provides the rationale for the decisions made

Risk Mitigation and Countermeasure Options

Common Weakness Risk Analysis Framework (CWRAF)

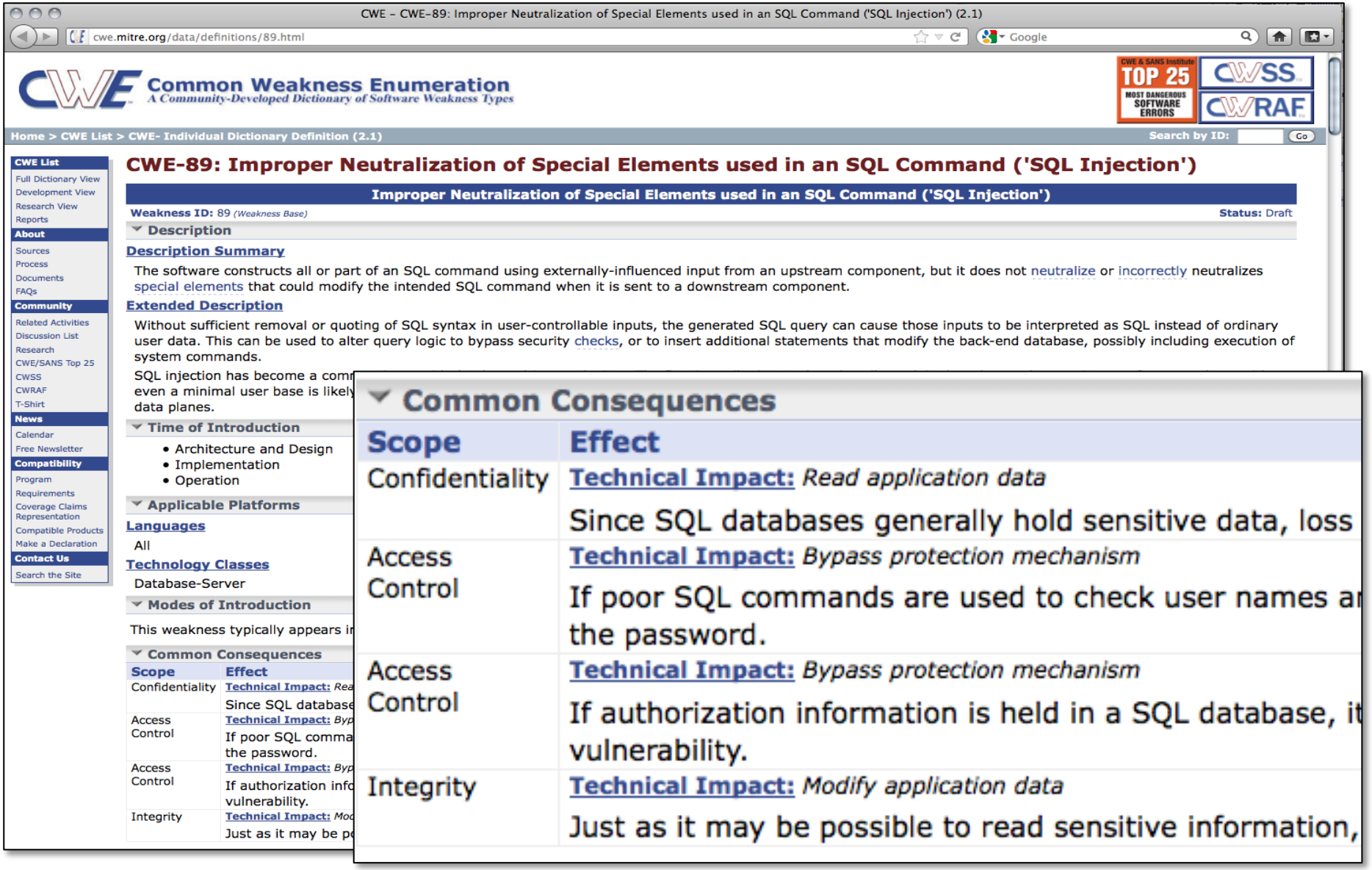
*How do I **identify** which of the 900+ CWE's are most important for my specific business domain, technologies and environment?*

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?

Technical Impacts – Common Consequences



Technical Impacts – Common Weakness Risk Analysis Framework (CWRAF)

1. Modify data
2. Read data
3. DoS: unreliable execution
4. DoS: resource consumption
5. Execute unauthorized code or commands
6. Gain privileges / assume identity
7. Bypass protection mechanism
8. Hide activities

Technical Impacts for CWE Entries

Note that this list is likely to change in future CWE versions.

CWE-89 (SQL Injection) has three technical impacts as listed in the Common_Consequences element of the CWE entry:

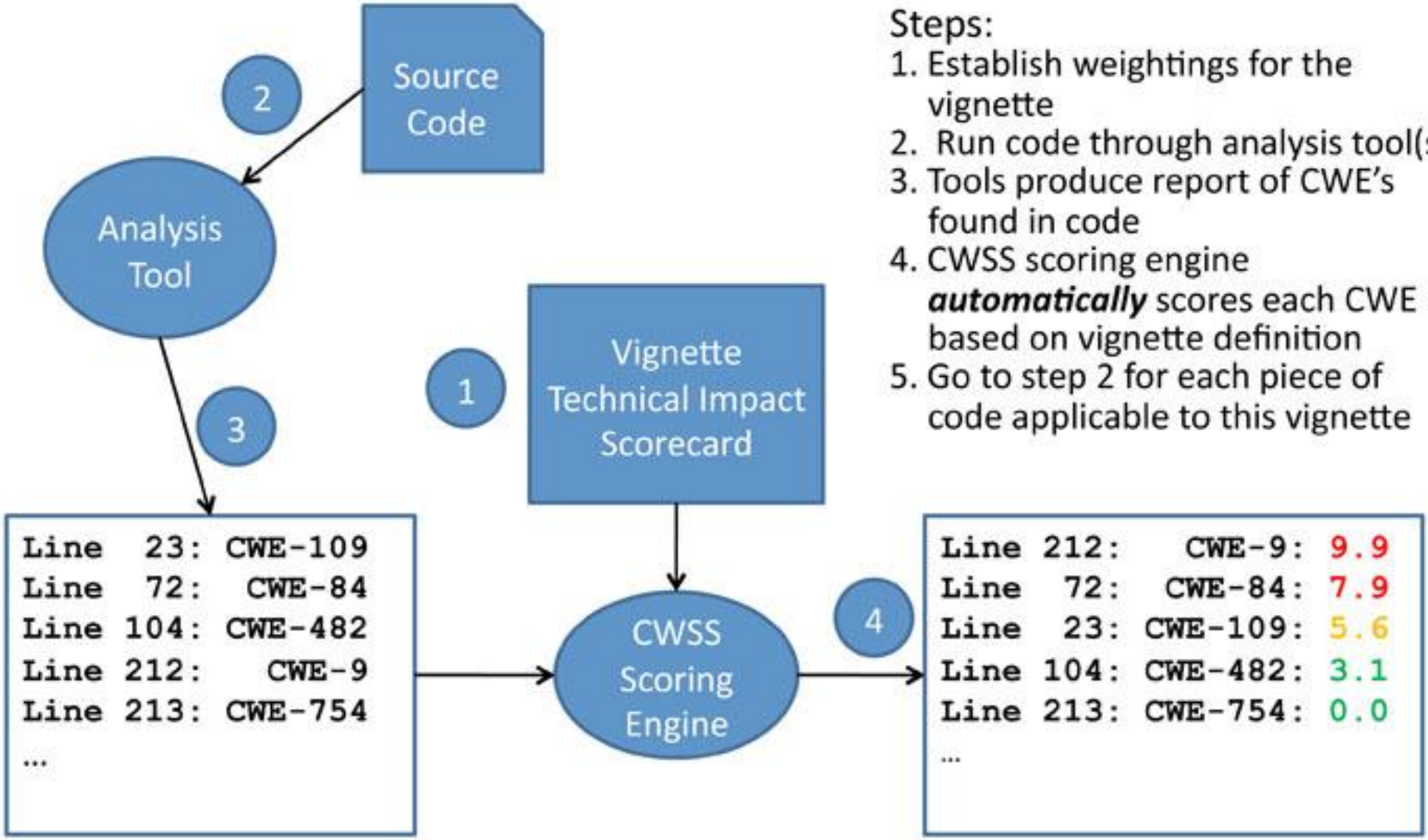
- Read application data
- Modify application data
- Bypass protection mechanism

For CWE-120 (Classic Buffer Overflow), the listed technical impacts are:

- Execute unauthorized code or commands
- DoS: crash / exit / restart

ID	Name	Max Subscore	Technical Impacts and Importance Subscores
CWE-89	SQL Injection	8	* Read data (8) * Modify data (8) * Bypass protection mechanism (7)
CWE-120	Classic Buffer Overflow	10	* Execute unauthorized code or commands (10) * DoS: unreliable execution (4)

Scoring Weaknesses Discovered in Code using CWSS

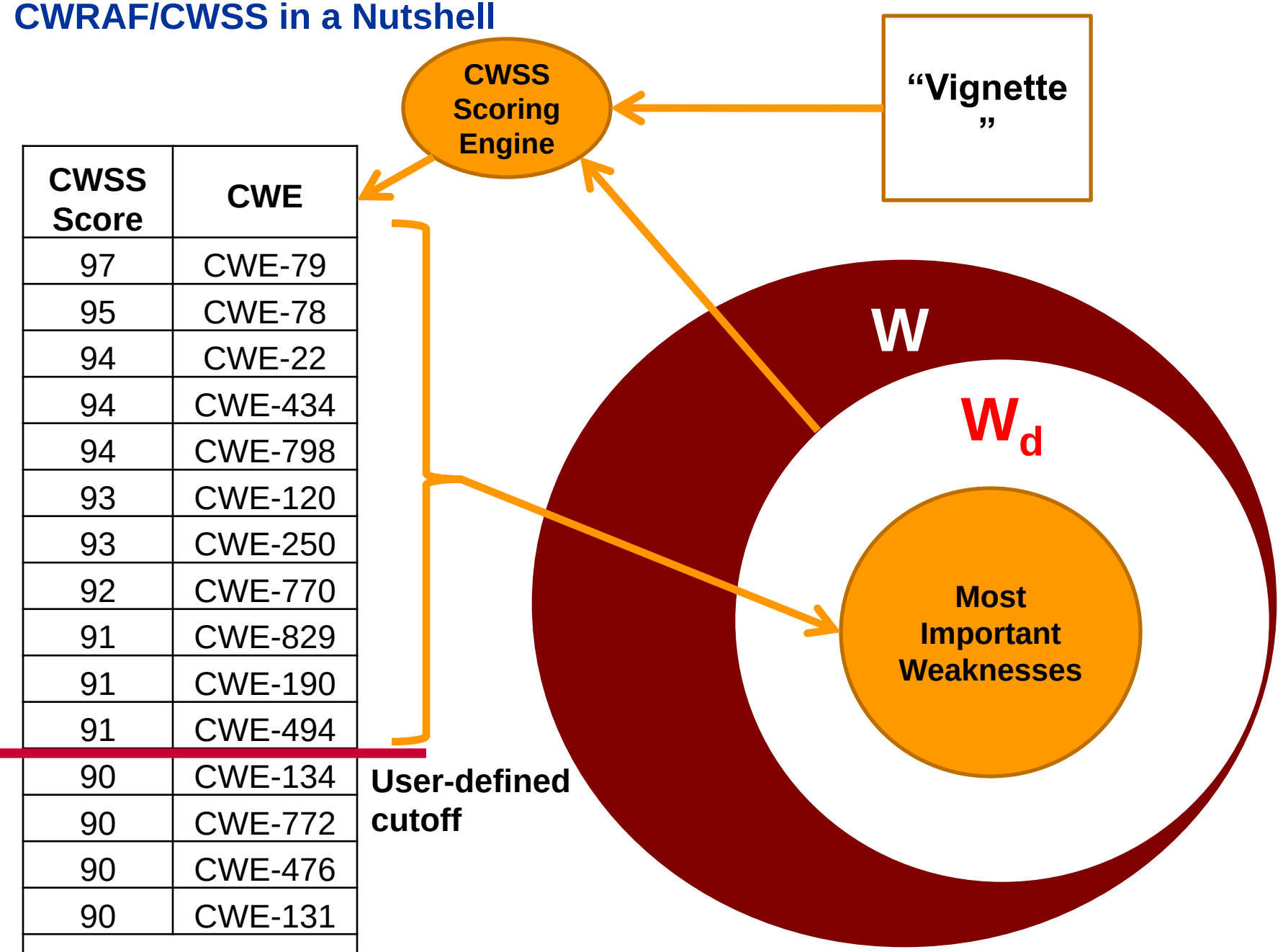


Steps:

1. Establish weightings for the vignette
2. Run code through analysis tool(s)
3. Tools produce report of CWE's found in code
4. CWSS scoring engine ***automatically*** scores each CWE based on vignette definition
5. Go to step 2 for each piece of code applicable to this vignette

Step 1 is only done once – the rest is automatic

CWRAF/CWSS in a Nutshell



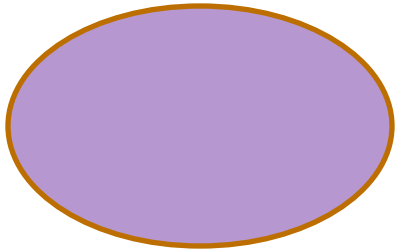
CWSS Score	CWE
97	CWE-79
95	CWE-78
94	CWE-22
94	CWE-434
94	CWE-798
93	CWE-120
93	CWE-250
92	CWE-770
91	CWE-829
91	CWE-190
91	CWE-494
90	CWE-134
90	CWE-772
90	CWE-476
90	CWE-131
…	

W is all possible weaknesses; Wd is all known weaknesses (CWE)

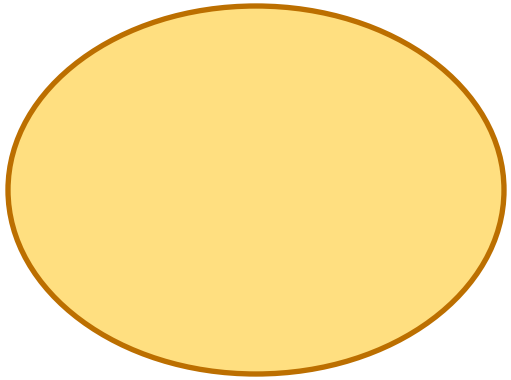
CWE Coverage Claims

Set of CWE's a capability *claims* to cover

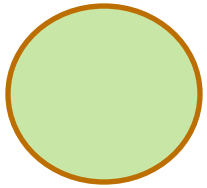
Tool A



Tool B



Pen Testing Service



Which static analysis tools and Pen Testing services find the CWE's I care about?

CWSS for a Technology Group

50%	Web Vignette 1 … TI(1), TI(2), TI(3),…	Top N List 1
10%	Web Vignette 2 … TI(1), TI(2), TI(3),…	Top N List 2
10%	Web Vignette 3 … TI(1), TI(2), TI(3),…	Top N List 3
10%	Web Vignette 4 … TI(1), TI(2), TI(3),…	Top N List 4
15%	Web Vignette 5 … TI(1), TI(2), TI(3),…	Top N List 5
15%	Web Vignette 6 … TI(1), TI(2), TI(3),…	Top N List 6
	Web Application Technology Group	**Top 10 List**

CWE Top 10 List for Web Applications can be used to:

- **Identify skill and training needs for your web team**
- **Include in T's & C's for contracting for web development**
- **Identify tool capability needs to support web assessment**

MITRE

CWE List
Full Dictionary View
Development View
Research View
Reports
About
Sources
Process
Documents
FAQs
Community
SwA On-Ramp
T-Shirt
Discussion List
Discussion Archives
Scoring
CWSS
CWRAF
CWE/SANS Top 25
Compatibility
Requirements
Coverage Claims Representation
Compatible Products
Make a Declaration
News
Calendar
Free Newsletter
Contact Us
Search the Site

Common Weakness Risk Analysis Framework (CWRAF™)

CWRAF provides a framework for scoring software weaknesses in a consistent, flexible, open manner, while accommodating context for the various business domains. It is a collaborative, community-based effort that is addressing the needs of its stakeholders across government, academia, and industry. CWRAF is a part of the Common Weakness Enumeration (CWE™) project, co-sponsored by the Software Assurance program in the National Cyber Security Division (NCSD) of the US Department of Homeland Security (DHS).

CWRAF benefits:

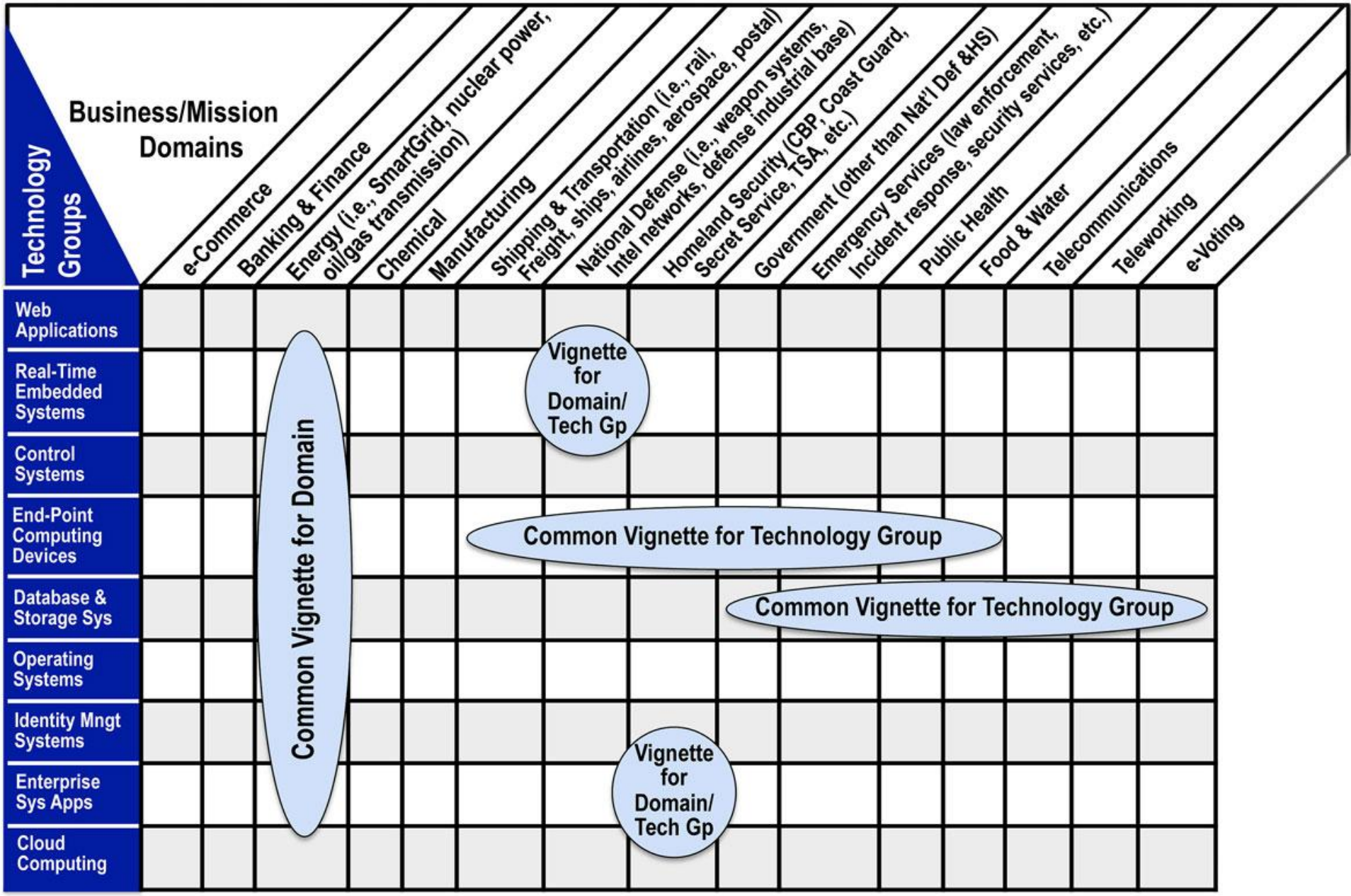
- Includes a mechanism for measuring risk of security errors ("weaknesses") in a way that is closely linked with the risk to an organization's business or mission.
- Supports the automatic selection and prioritization of relevant weaknesses, customized to the specific needs of the organization's business or mission.
- Can be used by organizations in conjunction with the Common Weakness Scoring System (CWSS™) to identify the most important weaknesses for their business domains, in order to inform their acquisition and protection activities as one part of the larger process of achieving software assurance.

CWRAF and CWSS allow users to rank classes of weaknesses independent of any particular software package, in order to prioritize them relative to each other (e.g., "buffer overflows are higher priority than memory leaks"). This approach, sometimes referred to as a "Top-N list," is used by the CWE/SANS Top 25, OWASP Top Ten, and similar efforts. CWRAF and CWSS allow users to create their own custom Top-N lists.

CWRAF Version 0.8.1

Section Contents
CWRAF
Introduction
CWSS Scoring in CWRAF
Creating Your Own Vignettes
Future Versions and Activities
Change Log
Vignettes
Tech Groups and Domains
Archetypes
CWRAF FAQs
Other Items of Interest
CWSS
Terms of Use

Vignettes – Technology Groups & Business/Mission Domains



Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups

Tech Groups / Business Domains	banking-finance	chemical	ecomm	emerg-svc	energy	evoting	human-res	natl-defense	pub-health	soc-media	telecom
Web Applications	fin-trade, e-banking		retail-www		smart-meter, smart-grid-RUS, smart-grid-gw, reg-elec, scada-hist, web-scada-hmi	elec-abs-int, evoting-Internet, corp-vote	emp-comp		med-billing, med-device	soc-net, elec-date	tel-ras, web-mail
Real-Time Embedded Systems					smart-meter, smart-grid-RUS, smart-grid-gw	evoting-DRE		weap-sensor	med-device		
Control Systems		chem-flow			smart-meter, smart-grid-RUS, smart-grid-gw, reg-elec, scada-hist, web-scada-hmi						
End-Point Computing Devices				first-resp					med-device		
Database & Storage Systems	e-banking		retail-www		scada-hist, web-scada-hmi	evoting-DRE	emp-comp		med-billing		
Operating Systems			retail-www		web-scada-hmi	elec-abs-int, evoting-Internet, corp-vote			med-billing, med-device		
Identity Management Systems											
Enterprise Systems & Applications	e-banking		retail-www		scada-hist, web-scada-hmi	elec-abs-int, evoting-Internet, corp-vote	emp-comp		med-billing, med-device		
Cloud Computing											
Enterprise Security Products											
Network Communications					web-scada-hmi	evoting-DRE, evoting-Internet					

Domain Summary

This is an up-to-date list of domains as used by CWRAF. For each domain, a list of associated vignettes is provided.

Domain	Description
e-Commerce	The use of the Internet or other computer networks for the sale of products and services, typically using the WWW. Vignettes: Web-Based Retail Provider
Banking & Finance	Financial industry, including depository financial institutions (banks, thrifts, and credit unions), insurers, securities brokers/dealers, investment companies, some financial utilities, and their associated regulatory systems and agencies. Vignettes: Financial Trading, Online Banking
Energy	Smart Grid (electrical network through a large region, using digital technology for monitoring or control), nuclear power stations, oil and gas transmission, etc. Vignettes: Household Smart Meter, Smart Grid remote utility server, Smart Grid Neighborhood Gateway, Regional Electricity Flow Control, SCADA Historian, Distributed Production Facility Management using SCADA Web-based HMI
Chemical	Chemical processing and distribution, etc. Vignettes: Chemical Flow Control
Manufacturing	Plants and distribution channels, supply chain, etc. *No vignettes defined.*
Shipping & Transportation	Aerospace (such as safety-critical ground aviation systems, on-board avionics, etc.), highway, maritime transportation, mass transit, pipeline systems, and rail. *No vignettes defined.*
National Defense	Weapon systems, Intel networks, Defense Industrial Base, etc. Vignettes: Weapon system sensor
Homeland Security	CBP, Coast Guard, Secret Service, TSA, etc. *No vignettes defined.*

Government (Other)	Government (other than National Defense and Homeland Security) *No vignettes defined.*
Emergency Services	Systems and services that support for First Responders, incident management and response, law enforcement, and emergency services for citizens, etc. The organizations and processes for protecting and preserving critical assets before, during, and after a disaster or catastrophe. Vignettes: First Responder
Public Health	Health care, medical encoding and billing, patient information/data, critical or emergency care, medical devices (implantable, partially embedded, patient care), drug development and distribution, food processing, clean water treatment and distribution (including dams and processing facilities), etc. Vignettes: Medical Billing, Human Medical Devices
Food & Water	Food processing, clean water treatment and distribution (including dams and processing facilities), etc. *No vignettes defined.*
Telecommunications	Cellular services, land lines, VOIP, cable & fiber networks, etc. Vignettes: Teleworking - Remote Access Server, Teleworking - Web Mail
Teleworking	Support for employees to have remote access to internal business networks and capabilities, e.g. networking-capable PDAs and cell phones, VPNs, Network Access Control (NAC), Web-based email services, etc. *No vignettes defined.*
e-Voting	Electronic voting systems, whether for state-run elections, shareholder meetings, etc. Vignettes: State Election Administration using remote Internet voting via absentee ballot, State or Local Elections using eVoting via Direct Recording Election Machines., State or Local Elections using eVoting via an Internet web application, Corporate Shareholder Internet voting
Social Media	(Example Domain) The use of the Internet or other computer networks for communication, collaboration, or entertainment in which a large group of users can interact with each other. This includes social networking, wikis, blogs, music and photograph sharing, product/service reviews, bookmarking, etc. Vignettes: Social Networking, Electronic Dating
Human Resources	(Example Domain) Human resources - management of personnel within an organization, including recruitment, compensation (salary and benefits), performance assessment, training, etc. Vignettes: Employee Compensation

Vignette Summary

banking-finance	
Financial Trading	Internet-facing, E-commerce provider of retail goods or services. Data-centric - Database containing PII, credit card numbers, and inventory.
Online Banking	The web-based interaction between a bank, credit union, or other financial institution and its consumers for managing accounts, paying bills, and conducting financial transactions.
chemical	
Chemical Flow Control	A SCADA-based flow control system for a chemical plant. Underlying technology - heavy C usage. Systems developed in pre-Internet era with management consoles interfacing to them.
ecomm	
Web-Based Retail Provider	Internet-facing, E-commerce provider of retail goods or services. Data-centric - Database containing PII, credit card numbers, and inventory.
emerg-svc	
First Responder	First responder (such as fire, police, and emergency medical personnel) for a disaster or catastrophe.
energy	
Household Smart Meter	Meter within the Smart Grid that records electrical consumption and communicates this information to the supplier on a regular basis.
Smart Grid remote utility server	Obtains information from smart meters through neighborhood gateways.
Smart Grid Neighborhood Gateway	Appliance between smart meter and remote utility server.
Regional Electricity Flow Control	Flow control for an electricity network throughout a relatively large region, to further connect suppliers and consumers. Power now enters the grid from both sides (classic provider, but also home-to-provider e.g. home photo-voltaic and wind turbines in homes and throughout the landscape). System needs to have "smarts" to the load leveling capabilities of the grid which is basically a large distributed SCADA-type system.
SCADA Historian	Historian server for archival and analysis of data for a SCADA system. Contains a database backend and is accessible via a web interface. Access to the server is typically restricted to a DMZ or internal network.
Distributed Production Facility Management using SCADA Web-based HMI	A web-based Human Machine Interface (HMI) for SCADA systems. Users can visualize and control industrial automation processes in real-time from a control interface directly in communication with remote sensors and data collection points. All facets of production can be monitored and managed from a web browser. The HMI uses various frameworks (Java, .NET, etc.) with Restful Architecture (AJAX, XML, SOAP, XSL, and WML).

evoting	
State Election Administration using remote Internet voting via absentee ballot	Internet-facing polling system supporting high-volume transactions, high availability, Data-centric Database containing ballot information, Audit log generation for each voter.
State or Local Elections using eVoting via Direct Recording Election Machines.	DRE systems are not directly connected with the Internet. Vote data is uploaded to a centralized server via modem. Election worker retrieves hardcopies of the voting record from the machine and delivers the printouts to election officials. DRE machines are programmed with firmware uploaded from a compact flash card. It is generally accepted that the computer used to upload the firmware to the flash card should not be connected to the Internet.
State or Local Elections using eVoting via an Internet web application	Internet-facing polling systems are connected to the Internet and are designed to support high-volume transactions and high availability. A Data-centric Database is used to collect ballot information, Audit logs are generated for each voter.
Corporate Shareholder Internet voting	Corporate Shareholder voting using remote Internet voting
human-res	
Employee Compensation	Product for managing employee salary and bonuses. PII includes salary, financial transaction (e.g. for direct deposit), social security number, home address, etc.
natl-defense	
Weapon system sensor	Sensor for a weapons system that is connected to the Global Information Grid (GIG).
pub-health	
Medical Billing	Medical encoding and billing. Data used includes Electronic Health Records (EHR), financial management, interactions with insurance companies.
Human Medical Devices	Medical devices - "implantable" or "partially embedded" in humans, as well as usage in clinic or hospital environments ("patient care" devices.) Includes items such as pacemakers and automatic drug delivery. Control or monitoring of the device might be performed by smartphones. The devices are not in a physically secured environment.
soc-media	
Social Networking	Web site for enabling a large community of people to post comments, create profiles, exchange messages or pictures, and join affiliation groups, e.g. Facebook, MySpace, Twitter, or LinkedIn. Free-form content, high connectivity between users, private messaging. Heavy Web 2.0 usage.
Electronic Dating	Web site for electronic dating. Users can create profiles with pictures, exchange private email, participate in discussion forums, perform searches. Heavy Web 2.0.
telecom	
Teleworking - Remote Access Server	Remote Access Server used to support employees working outside the enterprise, including teleworking/telecommuting.
Teleworking - Web Mail	Use of web-based email for remote access.

CWE List
Full Dictionary View
Development View
Research View
Reports
About
Sources
Process
Documents
FAQs
Community
SwA On-Ramp
T-Shirt
Discussion List
Discussion Archives
Scoring
CWSS
CWRAF
CWE/SANS Top 25
Compatibility
Requirements
Coverage Claims Representation
Compatible Products
Make a Declaration
News
Calendar
Free Newsletter
Contact Us
Search the Site

Creating Your Own Vignettes

Currently, there are approximately 20 Vignettes and Technical Scorecards, but anyone can create their own Vignette and its accompanying Technical Scorecard to identify which CWEs are most significant to their business and applications. This section will help guide you through that process.

One of the items found in these sample Vignettes is the "Archetypes". A list of the currently defined Archetypes that are available for use in describing Vignettes is here. If there are new Archetypes you need just identify them and send them to cwe@mitre.org and we can add them to the list.

These Archetypes are used as the context for describing the technical elements utilized by the application described in the Vignette.

There are two tables for each Vignette, "Vignette Definition" and "Technical Impact Scorecard".

Vignette Definition

Creating a Vignette Definition basically comes down to filling in the Vignette Definition table. Below is an example Vignette Definition table with a specific Vignette for a Web-Based Retail Provider described. The Vignette Definition is meant to talk about what business issues are of concern for the application. Is the application dealing with PII? Credit card (PCI-relevant) data? How bad is each of the 8 Technical Impacts given what the application is doing for a business (in the business's operational context).

Name	Web-Based Retail Provider
ID	retail-www
Maturity	under-development
Domain	ecomm
Desc	Internet-facing, E-commerce provider of retail goods or services. Data-centric - Database

Section Contents
CWRAF
Introduction
CWSS Scoring in CWRAF
Creating Your Own Vignettes
Future Versions and Activities
Change Log
Vignettes
Tech Groups and Domains
Archetypes
CWRAF FAQs
Other Items of Interest
CWSS
Terms of Use

CWRAF - Archetypes

Following is a list of the archetypes that are used in CWRAF.

Anti-Virus Program	
Authentication Server	Teleworking - Remote Access Server, Teleworking - Web Mail
B2B Communications	Medical Billing
Custom applications	
Database	Web-Based Retail Provider, Online Banking, Medical Billing, SCADA Historian, Distributed Production Facility Management using SCADA Web-based HMI, Employee Compensation
Development Framework	State or Local Elections using eVoting via an Internet web application
Digital certificate	
Distributed Control System (DCS)	
Document Processing	
Embedded Device	Human Medical Devices, Household Smart Meter, Smart Grid remote utility server, Smart Grid Neighborhood Gateway, State or Local Elections using eVoting via Direct Recording Election Machines., Weapon system sensor
Endpoint System	Distributed Production Facility Management using SCADA Web-based HMI, State or Local Elections using eVoting via Direct Recording Election Machines.
Firewall	
General-purpose OS	Web-Based Retail Provider, Medical Billing, Human Medical Devices, Distributed Production Facility Management using SCADA Web-based HMI, State Election Administration using remote Internet voting via absentee ballot, State or Local Elections using eVoting via an Internet web application, Corporate Shareholder Internet voting
Infrastructure as a Service (IaaS)	
Internet Communications	Distributed Production Facility Management using SCADA Web-based HMI, State or Local Elections using eVoting via an Internet web application
J2EE and supporting frameworks	Financial Trading
Laptop	
Modem Communications	State or Local Elections using eVoting via Direct Recording Election Machines.
N-tier distributed	Financial Trading
PDA	

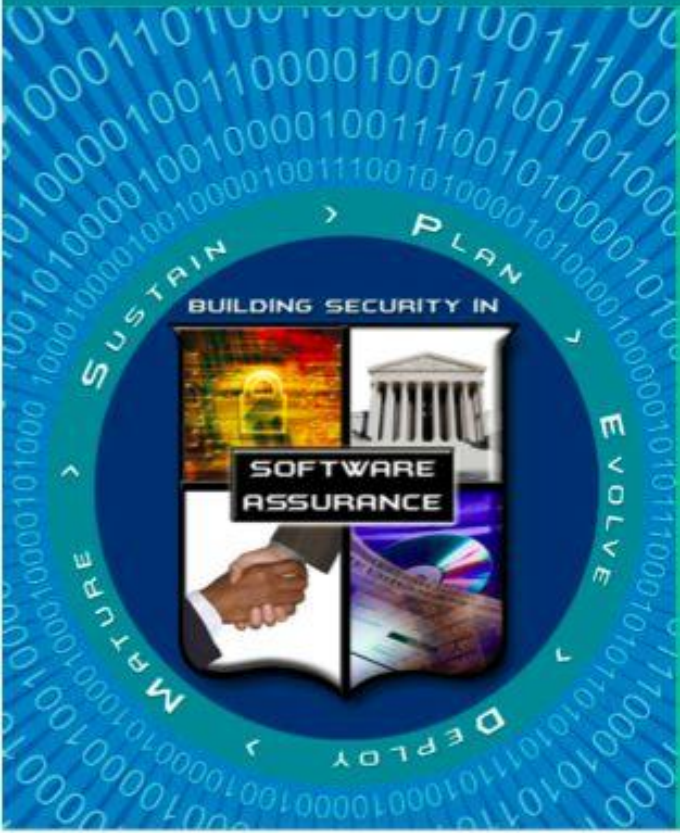
PKI	
Platform-as-a-Service (PaaS)	
Privacy management	
Process Control Systems	Household Smart Meter, Smart Grid remote utility server, Smart Grid Neighborhood Gateway, Regional Electricity Flow Control, SCADA Historian, Distributed Production Facility Management using SCADA Web-based HMI, Chemical Flow Control
Programmable Logic Controller (PLC)	
Proprietary Firmware	State or Local Elections using eVoting via Direct Recording Election Machines.
Remote Terminal Unit (RTU)	
Removable Storage Media	State or Local Elections using eVoting via Direct Recording Election Machines.
Router	
SCADA	
SOA-based web service	
Service-oriented architecture	Social Networking, Electronic Dating
Smartphone	Human Medical Devices, First Responder
Software-as-a-Service (SaaS)	
Transactional engine	Financial Trading, Online Banking
VPN	
Virtualized OS	
Web application	Distributed Production Facility Management using SCADA Web-based HMI, State or Local Elections using eVoting via an Internet web application
Web browser	Web-Based Retail Provider, Online Banking, Medical Billing, Distributed Production Facility Management using SCADA Web-based HMI, State Election Administration using remote Internet voting via absentee ballot, State or Local Elections using eVoting via an Internet web application, Corporate Shareholder Internet voting, Social Networking, Electronic Dating, Employee Compensation, Teleworking - Remote Access Server, Teleworking - Web Mail
Web browser plugin	
Web client	Human Medical Devices, Household Smart Meter, Smart Grid remote utility server, Smart Grid Neighborhood Gateway, Regional Electricity Flow Control, SCADA Historian
Web proxy	
Web server	Web-Based Retail Provider, Online Banking, Medical Billing, Regional Electricity Flow Control, SCADA Historian, Distributed Production Facility Management using SCADA Web-based HMI, State Election Administration using remote Internet voting via absentee ballot, Corporate Shareholder Internet voting, Social Networking, Electronic Dating, Employee Compensation, Teleworking - Remote Access Server, Teleworking - Web Mail
Web service	Distributed Production Facility Management using SCADA Web-based HMI

Software Assurance Automation

- Use cases for SwA Automation:
  - **SwA conditions/evidence for apps in an app store**
  - **SwA rating systems for determining which weaknesses are most important**
  - **Review/Discussion of the updated "Key Practices" Pocket Guide draft**
- Security automation standards in a cyber campaign and kill chain, as well as commercial offerings and operations and development

Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses

Software Assurance Pocket Guide Series:
Development, Volume II
Version 2.2, June 26, 2012 (Draft)



Software Assurance (SwA) Pocket Guide Resources

This is a resource for 'getting started' in selecting and adopting relevant practices for engineering, developing, and delivering secure software. As part of the Software Assurance (SwA) Pocket Guide series, this resource is offered for informative use only; it is not intended as directive or presented as being comprehensive since it references and summarizes material in the source documents and on-line resources that provide detailed information. When referencing any part of this document, please provide proper attribution and reference the source documents, when applicable.

This volume of the SwA Pocket Guide series focuses on key practices for mitigating the most egregious exploitable software weaknesses. It identifies mission/business risks attributable to the respective weaknesses, it identifies common attacks that exploit those weaknesses, and provides recommended practices for preventing the weaknesses. It provides insight for how software weaknesses are prioritized to guide training, development and procurement efforts.

At the back of this pocket guide are references, 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 http://buildsecurityin.us-cert.gov/swa .

Acknowledgements

The SwA Forum and Working Groups function as a stakeholder mega-community that welcomes additional participation in advancing software security and refining SwA-related information resources that are offered free for public use. Input to all SwA resources is encouraged. Please contact Software.Assurance@dhs.gov for comments and inquiries.

The SwA Forum is composed of government, industry, and academic members. The SwA Forum focuses on incorporating SwA considerations in educatoin, acquisition, and development processes relative to potential risk exposures that could be introduced by software and the software supply chain.

Participants in the SwA Forum's Processes & Practices Working Group collaborated with the Technology, Tools and Product Evaluation Working Group in developing the material used in this pocket guide as a step in raising awareness on how to incorporate SwA throughout the Software Development Life Cycle (SDLC).

Lacking common characterization of exploitable software constructs and how they could be attacked with associated mitigation practices previously presented one of the major challenges to realizing software assurance objectives. As part

- *"Fundamental Practices for Secure Software Development, 2ND EDITION, A Guide to the Most Effective Secure Development Practices in Use Today", SAFECode, February 8, 2011 at http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf*

Background

The 2011 CWE/SANS Top 25 Most Dangerous Programming Errors is a consensus list of the most significant programming errors that can lead to serious software vulnerabilities. They occur frequently, are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

The list is the result of collaboration between the MITRE CWE team, many top software security experts in the US and Europe, and the SANS Institute. It leverages experiences in the development of the SANS Top 20 attack vectors (http://www.sans.org/top20/), MITRE's Common Weakness Enumeration (CWE) (http://cwe.mitre.org/), and MITRE's Common Attack Pattern Enumeration and Classification (CAPEC) (https://capec.mitre.org/). With the sponsorship and support of the US Department of Homeland Security's National Cyber Security Division Software Assurance Program, MITRE maintains the CWE and CAPEC websites, presenting detailed descriptions of the top 25 programming errors along with authoritative guidance for mitigating and avoiding them. The CWE site also contains data on more than 800 additional programming errors, design errors, and architecture errors that can lead to exploitable vulnerabilities. See CWE Frequently Asked Questions at http://cwe.mitre.org/about/faq.html .

A goal for the CWE Top 25 list is to stop vulnerabilities at the source by educating programmers on how to eliminate all-too-common mistakes before software is even shipped. The list serves as a tool for education and awareness to help programmers prevent the kinds of vulnerabilities that plague the software industry. Software consumers can use the same list to help them to ask for more secure software. Finally, software managers, testers, and CIOs can use the CWE Top 25 list as a means for selecting the best tools and services for their needs and as a measuring stick of progress in their efforts to secure their software.

Top 25 Common Weaknesses

Table 1 provides the Top 25 CWEs organized into three high-level categories that contain multiple CWE entries:

1. Insecure Interaction Between Components
2. Risky Resource Management
3. Porous Defenses

Table 1 - Top 25 Common Weakness Enumeration (CWE)

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

CWE	Description
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection').
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting').
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection').
CWE-352	Cross-Site Request Forgery (CSRF).
CWE-434	Unrestricted Upload of File with Dangerous Type.
CWE-601	URL Redirection to Untrusted Site ('Open Redirect').

Risky Resource Management

These weaknesses are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

CWE	Description
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal').
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow').
CWE-131	Incorrect Calculation of Buffer Size.
CWE-134	Uncontrolled Format String.
CWE-190	Integer Overflow or Wraparound.
CWE-494	Download of Code Without Integrity Check.
CWE-676	Use of Potentially Dangerous Function.
CWE-829	Inclusion of Functionality from Untrusted Control Sphere.

Porous Defenses

These weaknesses are related to defensive techniques that are often misused, abused, or just plain ignored.

CWE	Description
CWE-250	Execution with Unnecessary Privileges.
CWE-306	Missing Authentication for Critical Function.
CWE-307	Improper Restriction of Excessive Authentication Attempts.
CWE-311	Missing Encryption of Sensitive Data.
CWE-327	Use of a Broken or Risky Cryptographic Algorithm.
CWE-732	Incorrect Permission Assignment for Critical Resource.
CWE-759	Use of a One-Way Hash without a Salt.
CWE-798	Use of Hard-coded Credentials.
CWE-807	Reliance on Untrusted Inputs in a Security Decision.
CWE-862	Missing Authorization.
CWE-863	Incorrect Authorization.

Selection of the Top 25 CWEs

The Top 25 CWE list was first developed at the end of 2008 and is updated on a yearly basis. Approximately 40 software security experts provided feedback, including software developers, scanning tool vendors, security consultants, government representatives, and university professors. Representation was international. Intermediate versions were created and resubmitted to the reviewers before the list was finalized. More details are provided in the Top 25 Process page at http://cwe.mitre.org/top25/process.html .

To help characterize and prioritize entries in the Top 25 CWE list, a threat model was developed that identified an attacker with solid technical skills and determined enough to invest some time into attacking an organization. Weaknesses in the Top 25 were selected using two primary criteria:

- **Weakness Prevalence:** how often the weakness appears in software that was not developed with security integrated into the software development life cycle (SDLC).
- **Consequences:** the typical consequences of exploiting a weakness if it is present, such as unexpected code execution, data loss, or denial of service.

Prevalence was determined based on estimates from multiple contributors to the Top 25 list, since appropriate statistics were not readily available.

With these criteria, future versions of the Top 25 CWEs will evolve to cover different weaknesses. Other CWEs that represent significant risks were listed as being on the cusp, and they can be viewed at http://cwe.mitre.org/ .

Information about the Weaknesses

The primary audience for CWE information is intended to be software programmers and designers. For each individual CWE entry, additional information is provided.

- CWE ID and name.
- Supporting data fields: supplementary information about the weakness that may be useful for decision-makers to further prioritize the entries.
- Discussion: Short, informal discussion of the nature of the weakness and its consequences.
- Prevention and Mitigations: steps that developers can take to mitigate or eliminate the weakness. Developers may choose one or more of these mitigations to fit their own needs. Note that the effectiveness of these techniques vary, and multiple techniques may be combined for greater defense-in-depth.
- Related CWEs: other CWE entries that are related to the Top 25 weakness. Note: This list is illustrative, not comprehensive.
- Related Attack Patterns: CAPEC entries for attacks that may be successfully conducted against the weakness. Note: the list is not necessarily complete.

See http://cwe.mitre.org for the additional supporting information on each CWE.

Other Supporting Data Fields in CWEs

Each Top 25 entry includes supporting data fields for weakness prevalence and consequences. Each entry also includes the following data fields.

- **Attack Frequency:** how often the weakness occurs in vulnerabilities that are exploited by an attacker.
- **Ease of Detection:** how easy it is for an attacker to find this weakness.
- **Remediation Cost:** the amount of effort required to fix the weakness.
- **Attacker Awareness:** the likelihood that an attacker is going to be aware of this particular weakness, methods for detection, and methods for exploitation.

Associated Mission/Business Risks and Related Attack Patterns

For each common weakness in software, there are associated risks to the mission or business enabled by the software. Moreover, there are common attack patterns that exploit those weaknesses.

Attack patterns are powerful mechanisms that capture and communicate the attacker's perspective. They are descriptions of common methods for exploiting software. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples. To assist in enhancing security throughout the software development lifecycle, and to support the needs of developers, testers and educators, the **CWE and Common Attack Pattern Enumeration and Classification (CAPEC)** are co-sponsored by DHS National Cyber Security Division as part of the Software Assurance strategic initiative, and the efforts are managed by MITRE. The CAPEC website provides a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. CAPEC will continue to evolve with public participation and contributions to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community.

Development teams should use attack patterns to understand the resilience of their software relative to common attacks and misuse. Table 2 lists the Mission/Business risks associated with each CWE, and it lists some of the possible attacks and misuses associated with the relevant CWEs which enable exploitation of the software.

For a full listing and description of all the attacks related to a particular CWE visit the websites for CWE and CAPEC at http://cwe.mitre.org and http://capec.mitre.org .

Table 2 - CWEs and Their Related Attack Patterns and Mission/Business Risks

CWE	Related Attack Pattern	Mission/Business Risks
CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	» CAPEC-23: File System Function Injection, Content Based » CAPEC-64: Using Slashes and URL Encoding Combined to Bypass Validation Logic » CAPEC-76: Manipulating Input to File System Calls » CAPEC-78: Using Escaped Slashes in Alternate Encoding » CAPEC-79: Using Slashes in Alternate Encoding » CAPEC-139: Relative Path Traversal	» DoS: crash / exit / restart » Execute unauthorized code or commands » Modify files or directories » Read files or directories
CWE-78 : Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	» CAPEC-6: TCP Header » CAPEC-15: Command Delimiters » CAPEC-43: Exploiting Multiple Input Interpretation Layers » CAPEC-88: OS Command Injection » CAPEC-108: Command Line Execution through SQL Injection	» DoS: crash / exit / restart » Execute unauthorized code or commands » Hide activities » Modify application data » Modify files or directories » Read application data » Read files or directories
CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	» CAPEC-18: Embedding Scripts in Nonscript Elements » CAPEC-19: Embedding Scripts within Scripts » CAPEC-32: Embedding Scripts in HTTP Query Strings » CAPEC-63: Simple Script Injection » CAPEC-85: Client Network Footprinting (using AJAX/XSS) » CAPEC-86: Embedding Script (XSS) in HTTP Headers » CAPEC-91: XSS in IMG Tags » CAPEC-106: Cross Site Scripting through Log Files » CAPEC-198: Cross-Site Scripting in Error Pages » CAPEC-199: Cross-Site Scripting Using Alternate Syntax » CAPEC-209: Cross-Site Scripting Using MIME Type Mismatch » CAPEC-232: Exploitation of Privilege/Trust » CAPEC-243: Cross-Site Scripting in Attributes » CAPEC-244: Cross-Site Scripting via Encoded URI Schemes	» Bypass protection mechanism » Execute unauthorized code or commands » Read application data

Table 2 - CWEs and Their Related Attack Patterns and Mission/Business Risks		
CWE	**Related Attack Pattern**	**Mission/Business Risks**
	» CAPEC-184: Software Integrity Attacks » CAPEC-185: Malicious Software Download » CAPEC-193: PHP Remote File Inclusion » CAPEC-222: iFrame Overlay » CAPEC-251: Local Code Inclusion » CAPEC-252: PHP Local File Inclusion » CAPEC-253: Remote Code Inclusion	
CWE-862 : Missing Authorization	» CAPEC-1: Accessing Functionality Not Properly Constrained by ACLs » CAPEC-17: Accessing, Modifying or Executing Executable Files » CAPEC-58: Restful Privilege Elevation » CAPEC-122: Exploitation of Authorization » CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels	» Bypass protection mechanism » Gain privileges / assume identity » Modify application data » Modify files or directories » Read application data » Read files or directories
CWE-863 : Incorrect Authorization	» CAPEC-1: Accessing Functionality Not Properly Constrained by ACLs » CAPEC-17: Accessing, Modifying or Executing Executable Files » CAPEC-58: Restful Privilege Elevation » CAPEC-122: Exploitation of Authorization » CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels	» Bypass protection mechanism » Gain privileges / assume identity » Modify application data » Modify files or directories » Read application data » Read files or directories

Key Practices

The key practices documented in "2011 CWE/SANS Top 25 Most Dangerous Programming Errors" focus on preventing and mitigating dangerous programming errors. Some of the Key Practices specified in the pocket guide are derived from mitigation recommendations that were common across many of the CWEs in the CWE Top 25, and others came from approaches described on the CERT Secure Coding Wiki. Additional information on preventing the various weaknesses is available in the CERT Secure Coding Wiki at https://www.securecoding.cert.org/ and other websites listed under On-Line Resources of this SwA Pocket Guide. Development teams are also encouraged to use the CAPEC attack patterns to gain understanding of how their software can be attacked, as well as considering how they can engineer their software to better handle such attacks. They are also encouraged to use the CAPEC attack patterns to develop tests that can determine the resilience of their code relative to the common attacks used to exploit software weaknesses. In this SwA Pocket Guide the key practices are grouped in tables according to Software Development Life Cycle (SDLC) phases:

1. Requirements, Architecture, and Design (Table 3) ;
2. Build, Compilation, Implementation, Testing, and Documentation (Table 4) ;
3. Installation, Operation and System Configuration (Table 5) , and
4. Associated CERT Coding Rules (Table 6) .

Table 3 – Requirements, Architecture, and Design	
Prevention and Mitigation Practices	**CWE**
For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.	CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.	
When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs. For example, ID 1 could map to "inbox.txt" and ID 2 could map to "profile.txt". Features such as the ESAPI AccessReferenceMap provide this capability.[R.22.3]	
If at all possible, use library calls rather than external processes to recreate the desired functionality.	CWE-78 : Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
For any data that will be used to generate a command to be executed, keep as much of that data out of external control as possible. For example, in web applications, this may require storing the data locally in the session's state instead of sending it out to the client in a hidden form field.	
For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.	
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using the ESAPI Encoding control [R.78.8] or a similar tool, library, or framework. These will help the programmer encode outputs in a manner less prone to error.	
If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Some languages offer multiple functions that can be used to invoke commands. Where possible, identify any function that invokes a command shell using a single string, and replace it with a function that requires individual arguments. These functions typically perform appropriate quoting and filtering of arguments. For example, in C, the system() function accepts a string that contains the entire command to be executed, whereas execl(), execve(), and others require an array of strings, one for each argument. In Windows, CreateProcess() only accepts one command at a time. In Perl, if system() is provided with an array of arguments, then it will quote each of the arguments.	

Table 3 – Requirements, Architecture, and Design

Prevention and Mitigation Practices	CWE
For example, consider using authorization frameworks such as the JAAS Authorization Framework [R.862.5] and the OWASP ESAPI Access Control feature [R.862.4].	
For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page. One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.	
Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) [R.863.1] to enforce the roles at the appropriate boundaries. Note that this approach may not protect against horizontal authorization, i.e., it will not protect a user from attacking others with the same role.	CWE-863 : Incorrect Authorization
Ensure that you perform access control checks related to your business logic. These checks may be different than the access control checks that you apply to more generic resources such as files, connections, processes, memory, and database records. For example, a database may restrict access for medical records to a specific database user, but each record might only be intended to be accessible to the patient and the patient's doctor.	
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using authorization frameworks such as the JAAS Authorization Framework [R.863.4] and the OWASP ESAPI Access Control feature [R.863.5].	
For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page. One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.	

Table 4 – Build, Compilation, Implementation, Testing, and Documentation

Prevention and Mitigation Practices	CWE
Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue."	CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Table 5 – Installation, Operation and System Configuration

Prevention and Mitigation Practices	CWE
Use a feature like Address Space Layout Randomization (ASLR). [R.120.5] [R.120.7]	CWE-120 : Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
Use a CPU and operating system that offers Data Execution Protection (NX) or its equivalent. [R.120.7] [R.120.9]	
Use a feature like Address Space Layout Randomization (ASLR). [R.131.3][R.131.5]	CWE-131 : Incorrect Calculation of Buffer Size
Use a CPU and operating system that offers Data Execution Protection (NX) or its equivalent. [R.131.4][R.131.5]	
Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.	CWE-601 : URL Redirection to Untrusted Site ('Open Redirect')
For all configuration files, executables, and libraries, make sure that they are only readable and writable by the software's administrator.	CWE-732 : Incorrect Permission Assignment for Critical Resource
Do not assume that the system administrator will manually change the configuration to the settings that you recommend in the manual.	
Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.	CWE-829 : Inclusion of Functionality from Untrusted Control Sphere

Table 6 – Associated CERT Coding Rules

Prevention and Mitigation Practices	CWE
FIO02-C: Canonicalize path names originating from untrusted sources	CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
FIO02-CPP: Canonicalize path names originating from untrusted sources	
ENV03-C: Sanitize the environment when invoking external programs	CWE-78 : Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
ENV04-C: Do not call system() if you do not need a command processor	
STR02-C: Sanitize data passed to complex subsystems	
IDS07-J: Do not pass untrusted, unsanitized data to the Runtime.exec() method	
STR02-CPP: Sanitize data passed to complex subsystems	
ENV03-CPP: Sanitize the environment when invoking external programs	
ENV04-CPP: Do not call system() if you do not need a command processor	
No associated CERT coding rules listed for this CWE entry.	CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
No associated CERT coding rules listed for this CWE entry.	CWE-89 : Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Table 6 – Associated CERT Coding Rules	
Prevention and Mitigation Practices	**CWE**
No associated CERT coding rules listed for this CWE entry.	CWE-434 : Unrestricted Upload of File with Dangerous Type
SEC06-J: Do not rely on the default automatic signature verification provided by	CWE-494 : Download of Code Without Integrity Check
No associated CERT coding rules listed for this CWE entry.	CWE-601 : URL Redirection to Untrusted Site ('Open Redirect')
ERR07-C: Prefer functions that support error checking over equivalent functions that	CWE-676 : Use of Potentially Dangerous Function
FIO01-C: Be careful using functions that use file names for identification	
INT06-C: Use strtol() or a related function to convert a string token to an integer	
INT06-CPP: Use strtol() or a related function to convert a string token to an integer	
FIO01-CPP: Be careful using functions that use file names for identification	
FIO03-J: Create files with appropriate access permission	CWE-732 : Incorrect Permission Assignment for Critical Resource
SEC01-J: Do not allow tainted variables in privileged blocks	
ENV03-J: Do not grant dangerous combinations of permissions	
FIO06-CPP: Create files with appropriate access permissions	
FIO06-C: Create files with appropriate access permissions	
No associated CERT coding rules listed for this CWE entry.	CWE-759 : Use of a One-Way Hash without a Salt
MSC03-J: Never hard code sensitive information	CWE-798 : Use of Hard-coded Credentials
ENV03-CPP: Sanitize the environment when invoking external programs	CWE-807 : Reliance on Untrusted Inputs in a Security Decision
SEC09-J: Do not base security checks on untrusted sources	
No associated CERT coding rules listed for this CWE entry.	CWE-829 : Inclusion of Functionality from Untrusted Control Sphere
No associated CERT coding rules listed for this CWE entry.	CWE-862 : Missing Authorization
No associated CERT coding rules listed for this CWE entry.	CWE-863 : Incorrect Authorization

Table 7 - Shared Mitigations	
Mitigation	**CWE Entries**
MIT-10	Run or compile your software using features or extensions that automatically provide a protection mechanism that mitigates or eliminates buffer overflows. For example, certain compilers and extensions provide automatic buffer overflow detection mechanisms that are built into the compiled code. Examples include the Microsoft Visual Studio /GS flag, Fedora/Red Hat FORTIFY_SOURCE GCC flag, StackGuard, and ProPolice. 1. CWE-120 : Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') 2. CWE-131 : Incorrect Calculation of Buffer Size
MIT-11	Use a feature like Address Space Layout Randomization (ASLR).[R.XX.A] [R.XX.B]

Table 7 - Shared Mitigations	
Mitigation	**CWE Entries**
	1. CWE-131 : Incorrect Calculation of Buffer Size 2. CWE-190 : Integer Overflow or Wraparound
MIT-9	Consider adhering to the following rules when allocating and managing an application's memory: 1. CWE-120 : Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

Creating Custom Top-N Lists using CWSS and CWRAF

The CWE/SANS Top 25 is a great starting point, but each organization has its own set of business priorities, threat environment, and risk tolerance and for those with the an understanding of those issues, a more refined and custom Top 25 for their business and what software is doing for their business is possible through the Common Weakness Scoring System (CWSS) (https://cwe.mitre.org/cwss/) and the Common Weakness Risk Analysis Framework (CWRAF) (https://cwe.mitre.org/cwraf/). The mechanisms in CWSS and CWRAF minimize this difficulty by letting organizations model their own business impact considerations into a risk-scoring mechanism.

CWSS provides the mechanism for scoring software's weaknesses in a consistent, flexible, open manner while considering the context and reflecting the weaknesses' impacts against that context. It aims to provide a consistent approach for tools and services prioritizing their static- and dynamic-analysis findings while addressing government, academia, and industry stakeholder needs.

CWRAF uses the core scoring mechanisms from CWSS to let software developers and consumers prioritize their own target list of software weaknesses across their unique portfolio of software applications and projects, focusing on those with the greatest potential to harm their business. To reduce risk, organizations can select appropriate tools and techniques, focus staff training, and define contracting details to ensure outsourced efforts also address the prioritized issues.

CWRAF and CWSS let users create top-n lists for their particular software and business domains, missions, and technology groups. In conjunction with other activities, CWSS and CWRAF help developers and consumers introduce more robust and resilient software into their operational environments.

Key Discussion Points Between Developers and Consumers, Acquirers, and Project Management

Improving software assurance requires an honest dialog between consumers, acquirers, project managers, and developers on an ongoing basis. Here are some discussion points you can bring up to spark a discussion that will hopefully provide you and them a better understanding of what you and they are doing and need to do to help improve the assurance around your software.

1. Design/Development Practices
    a. Which BSIMM or OpenSAMM activities/practices are followed?
    b. Which SDLC activities are used to directly prevent or mitigate vulnerabilities in the application software? (e.g. threat modeling, automated code analysis (static or dynamic), etc).
    c. Which security controls have been utilized to mitigate specific problems (e.g. authentication, authorization, cryptography)
    d. Which security-related frameworks are used, such as ESAPI or built-in frameworks?

e. Which secure coding rules/practices are followed? (e.g. CERT, MISRA, ISO SC-22, custom). How is conformance enforced (e.g. automated tools during checkin)?

f. What differences, if any, exist between the secure development practices for legacy code, versus newly-developed code?

g. For each implemented IETF RFC, how are the concerns in the RFC's "Security" section mitigated?

h. Which "Top N" vulnerability/attack lists do your development practices actively attempt to address (CWE Top 25, OWASP Top Ten, custom Top-N list)?

2. Third-Party Software Management

a. Which third-party libraries are used by the software?

b. How does the development team keep current with third- party libraries so that it does not use code with known vulnerabilities?

c. How are third-party code changes and vulnerabilities tracked/monitored?

d. Which third-party libraries were independently examined for vulnerabilities before being included in the software?

3. Detection and Analysis

a. Which standardized analysis/testing methodologies are used to evaluate the software? (e.g. OWASP ASVS, OSSTMM)

b. Has an independent 3rd-party review been performed against the software? Did the review cover code implementation, design, architecture, or installation settings?

d. What tools are used for automated code analysis? Static or dynamic? White box or black box?

e. Which manual analysis techniques were used?

f What specifications, data formats, and protocols are used? Were any test case suites or fuzzing tools used to evaluate the implementation (e.g. PROTOS)?

g. What is the attack surface of the software (in privileged code and overall)?. What metrics are used? Can the attack surface be described in terms of CAPEC?

h. Which parts of the code have been most recently reviewed?

i. Which parts of the software contain legacy code whose analysis has been skipped?

4. Compiler/Environment

a. Which compiler settings are used to reduce or eliminate risk for key weaknesses (e.g. /GS switch)?

b. Were any compiler warnings ignored when compiling the code? If so, which ones and why?

c. Was the code compiled using safe libraries?

d. Which OS features are used to reduce or eliminate the risk of important weaknesses (e.g. DEP, ASLR)?

5. Configuration/Installation

a. Is the product installed "secure by default"?

b. Is the product installed so that critical executables, libraries, configuration files, registry keys, etc. cannot be modified by untrusted parties?

c. Does the software run with limited privileges? If not, how is privilege separation performed?

d. How does the documentation cover security-relevant settings for administrators to use to lock down the software?

e. Does the software work under FDCC/USGCB configurations, and/or other secure configurations?

f. How does the software restrict access to network ports?

6. Vulnerability Response

a. Is a security response center set up to handle incoming vulnerability reports from external parties?

b. How easy is it for independent researchers and non-customers to report vulnerabilities?

c. Are emergency procedures in place to quickly fix issues that are first discovered being exploited in the wild?

d. Are procedures in place to handle when vulnerabilities are publicly disclosed without notifying the developer or giving sufficient time to produce a patch)?

e. Is there a sufficiently comprehensive set of information sources that are monitored for reported vulnerabilities in your own software, in third-party products, and competitor/analogous products?

f. When a new weakness is found by an outside party, how are the software and associated development practices reviewed and modified to ensure that similar weaknesses are also detected and removed?

7. Vulnerability Disclosure

a. How are consumers of the software notified about new vulnerabilities found in the code?

b. For vulnerabilities that are publicly disclosed by other parties without a patch, is there a policy to provide public commentary before a patch is available?

c. Which details are disclosed to customers? What is disclosed to the general public?

d. Are any credits or compensation provided to independent vulnerability researchers?

8. What kind of evidence or proof can be offered regarding these claims?

Using Tools and Other Capabilities to Identify the Top 25

Developers and third-party analysts can use CWE-compatible tools that can map to CWE items in the CWE Top 25. With the advancing maturity and increasing adoption of CWE, most vendors of software analysis tools and services express their findings of weaknesses in code, design, and architecture using CWE identifiers. This common language for expressing weaknesses has eliminated much of the ambiguity and confusion surrounding exactly what the tool or service has found. At the same time, different vendors take different approaches as to how they look for weaknesses and what weaknesses they look for. The CWE Coverage Claims Representation (CCR) is a means for software analysis vendors to convey to their customers exactly which CWE-identified weaknesses they claim to be able to locate in software. The word claim is emphasized since neither the CCR itself nor the CWE Compatibility Program verify or otherwise vet these statements of coverage. The CWE Effectiveness Program will eventually fulfill this role of verification.

The main goal of the CCR is to facilitate the communication of unambiguous statements of the intention of a tool or service to discover specific, CWE-identified weaknesses in software. These statements of claim are intended to allow the providers of software analysis tools and services and the consumers of those tools and services to share a single, unambiguous understanding of the scope of software weakness analysis. CCR wants users of tools and services to be aware and informed of the coverage of the tools and services they make use of in analyzing their software, and when specific classes of weaknesses or individual weaknesses are of specific concern, they can make sure their tools and services are at least trying to find them. Having a mis-match between an organization's focus and the capabilities of their tools and services is not something to be discovered after using and depending on them, but rather is something that should be addressed in the initial discussions and exploration of bringing those capabilities to bear for the organization.

It is anticipated that the CCR will also foster innovation in the technology of software analysis tools and services by allowing vendors to clearly state their intentions with respect to weakness discovery and understand more clearly when there is a need for targeting additional weaknesses to address their customer's concerns. Currently, a tool that does a very deep analysis on a small subset of the entire set of CWE-defined weaknesses may be judged as inadequate by potential customers since, by definition, it fails to discover a broad set of weaknesses. However, with the CCR, the tool provider could supply a CCR document for that tool, clearly setting expectations as to the set of weaknesses that the tool attempts to discover. Tool consumers could then evaluate tools based on what specific CWE-identified weaknesses those tools claim to discover and how that coverage fits within their needs, rather than comparing it to the entire set of CWE-defined weaknesses.

Conclusion

The Software Assurance Pocket Guide Series is developed in collaboration with the SwA Forum and Working Groups and provides summary material in a more consumable format. The series provides informative material for SwA initiatives that seek to reduce software vulnerabilities, minimize exploitation, and address ways to improve the routine development, acquisition and deployment of trustworthy software products. Together, these activities will enable more secure and reliable software that supports mission requirements across enterprises and the critical infrastructure.

For additional information or contribution to future material and/or enhancements of this pocket guide, please consider joining any of the SwA Working Groups and/or send comments to Software.Assurance@dhs.gov . SwA Forums are open to all participants and free of charge. Please visit https://buildsecurityin.us-cert.gov for further information.

No Warranty

Reprints

Software Assurance (SwA) Pocket Guide Series

SwA is primarily focused on software security and mitigating risks attributable to software; better enabling resilience in operations. SwA Pocket Guides are provided; with some yet to be published. All are offered as informative resources; not comprehensive in coverage. All are intended as resources for 'getting started' with various aspects of software assurance. The planned coverage of topics in the SwA Pocket Guide Series is listed:

SwA in Acquisition & Outsourcing

I. Contract Language for Integrating Software Security into the Acquisition Life Cycle

II. Software Supply Chain Risk Management & Due-Diligence

SwA in Development

I. Integrating Security into the Software Development Life Cycle

II. Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses

III. Risk-based Software Security Testing

IV. Requirements & Analysis for Secure Software

V. Architecture & Design Considerations for Secure Software

VI. Secure Coding & Software Construction

VII. Security Considerations for Technologies, Methodologies & Languages

SwA Life Cycle Support

I. SwA in Education, Training & Certification

II. Secure Software Distribution, Deployment, & Operations

III. Code Transparency & Software Labels

IV. Assurance Case Management

V. Assurance Process Improvement & Benchmarking

VI. Secure Software Environment & Assurance Ecosystem

SwA Measurement & Information Needs

I. Making Software Security Measurable

II. Practical Measurement Framework for SwA & InfoSec

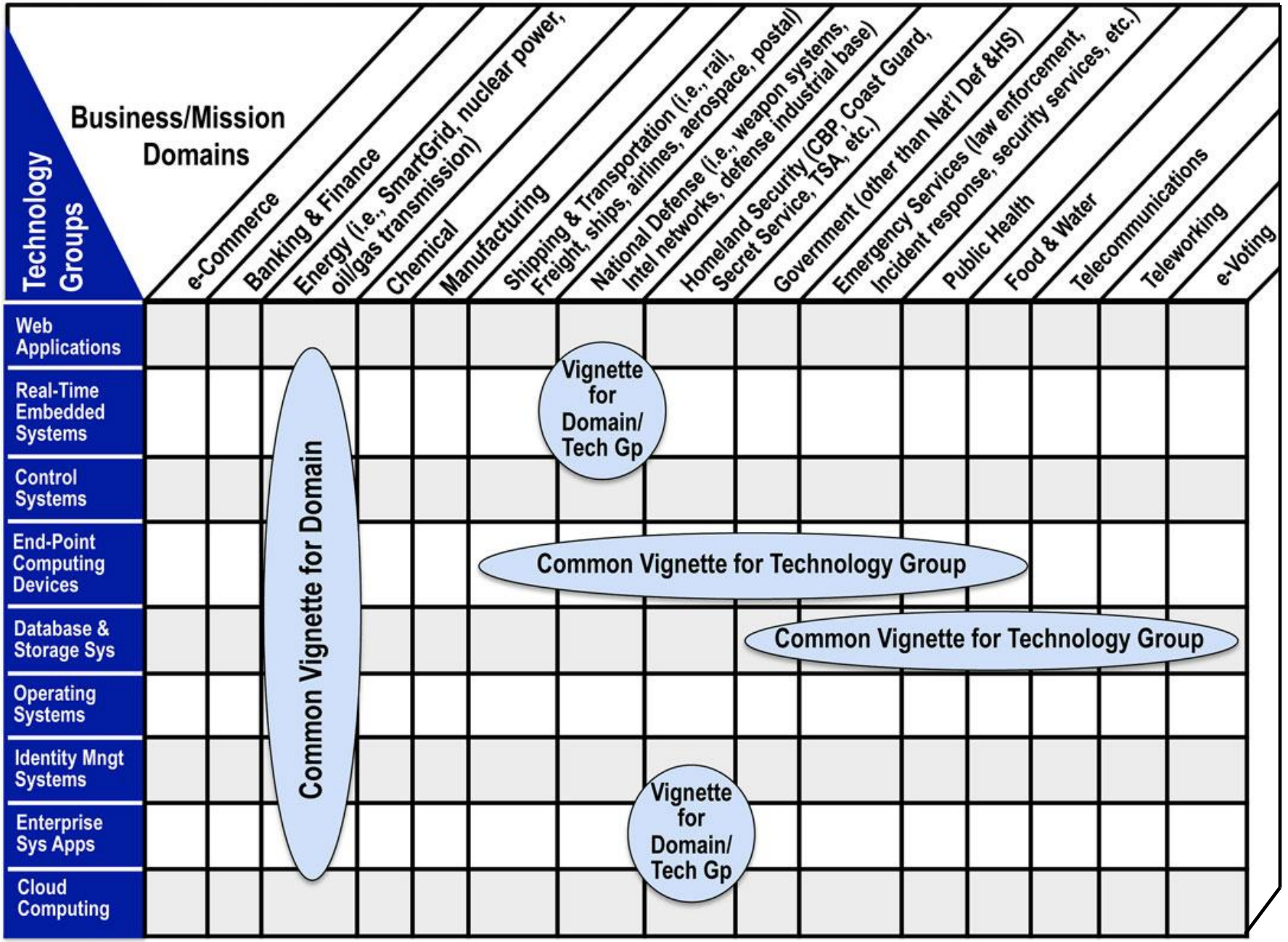
III. SwA Business Case & Return on Investment

SwA Pocket Guides and related documents are freely available for download via the DHS NCSD Software Assurance Community Resources and Information Clearinghouse at https://buildsecurityin.us-cert.gov/swa .

Software Assurance Automation

- Use cases for SwA Automation:
  - **SwA conditions/evidence for apps in an app store**
  - **SwA rating systems for determining which weaknesses are most important**
  - **Review/Discussion of the updated "Key Practices" Pocket Guide draft**
- Security automation standards in a cyber campaign and kill chain, as well as commercial offerings and operations and development

Leveraging Vignettes in Cyber Security Standardization for Key ICT Applications in various Domains

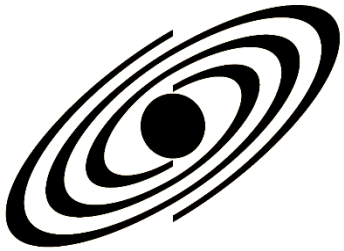


Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups

Common Weakness Risk Analysis Framework (CWRAF) and Common Weakness Scoring System (CWSS)

Organizations that have declared plans to work on CWRAF Vignettes and Technical Scorecards to help evolve CWRAF to meet their customer's and the community's needs for a scoring system for software errors.

Trustwave® SpiderLabs®



DTCC®

CISQ

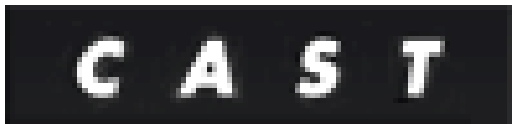
EC-Council

OWASP
The Open Web Application Security Project

SAIC®

Common Weakness Risk Analysis Framework (CWRAF) and Common Weakness Scoring System (CWSS)

Organizations that have declared plans to support CWSS in their future offerings and are working to help evolve CWSS to meet their customer's and the community's needs for a scoring system for software errors.



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.

Common Attack Pattern Enumeration and Classification (CAPEC)

Dictionary of attack types (mostly software)

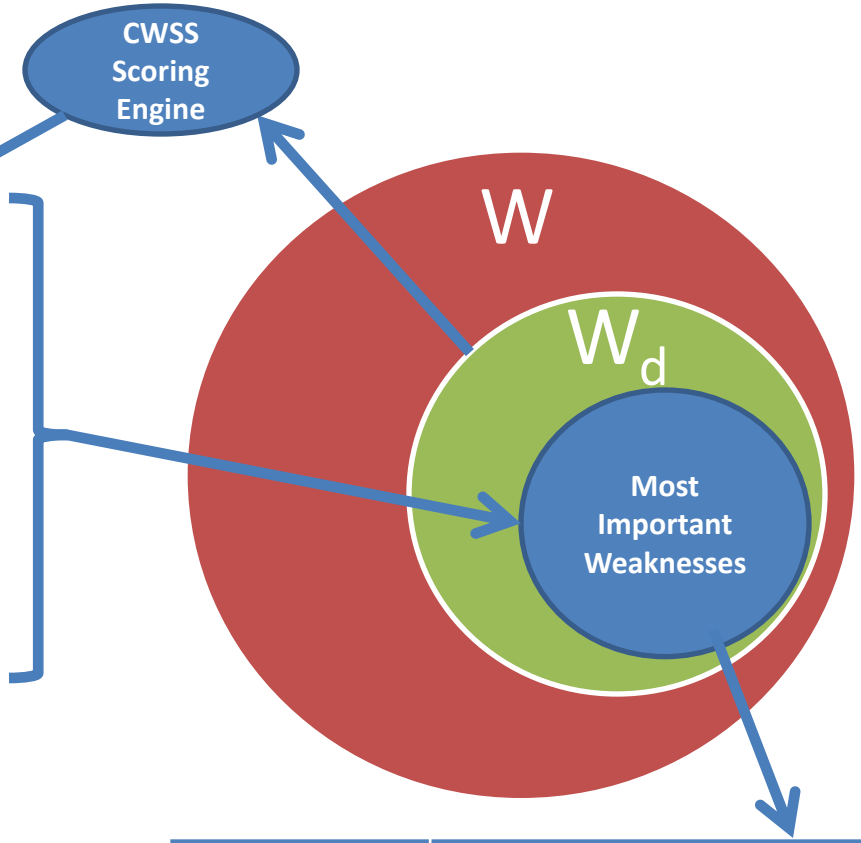
- CAPEC ID
- Name
- Description
- Attack Prerequisites
- Indicators of Attack
- Examples
- **Related Weaknesses (CWE's)**
- Mitigations

Plus much, much more

386 patterns, organized by categories, with views

What types of attacks should I test my system against?

CWSS Score	CWE
97	CWE-79
95	CWE-78
94	CWE-22
94	CWE-434
94	CWE-798
93	CWE-120
93	CWE-250
92	CWE-770
91	CWE-829
91	CWE-190
91	CWE-494
90	CWE-134
90	CWE-772
90	CWE-476
90	CWE-131
...	



CWE	Related CAPEC ID's
CWE-79	CAPEC-232, CAPEC-106, CAPEC-19, …
CWE-78	CAPEC-108, CAPEC-15, CAPEC-43, CAPEC-6, …
…	…

Common Attack Pattern Enumeration and Classification

Your Problem: How can you know that software you install or build is “secure”?

- Developers are often not trained in security, and thus let dangerous code slip into software.
- Most system owners likely have less understanding of these issues.

So, you need tools and techniques to help you find and eliminate weaknesses that might get built into your software with the developer workforce that you have.

Proposed Tools

- Lists of common weaknesses and attacks.
  - Tools built to cover these can help you know your staff is considering all known weaknesses.
  - Training developers about these can prevent weaknesses from being built in.
- Tools to check for weaknesses.
  - Running these on source code and/or compiled code can help you find weaknesses to remove.
  - Then, you can verify that they were removed.
- Metrics to assign priority to issues found.
  - These metrics can help you adjust your level of rigor to the risks and impact-level of the specific system.
  - Decide how much risk (if any) to accept.

Using the Tools

- Train developers to avoid the worst problems.
- Find security weaknesses in the code
  - Manual design/code review
  - Automated “static” analysis tools (for source code)
- Eliminate those weaknesses
- Test software against attacks
  - Automated “dynamic” application analysis tools (for executable code)
- Repeat as necessary

What software should you focus on?

- Sadly, attackers can attack even “unimportant” software on a machine or network to get to more important (but less vulnerable software).
- So, to some adequate extent, you must protect even your low-impact systems, to protect your higher impact systems/information.

Lists of common weaknesses and attacks: Common Weakness Enumeration (CWE)

- **Source:** Academia, private sector, public sector.
- **Content:** Dictionary of over 600 ***types*** of common software weaknesses.
  - Some examples: buffer overflow, command/SQL injection, missing/weak authentication, etc.
  - Typical info: code examples, common mitigation examples, links to related attacks
- **Typical Use:** COTS vendors require SW developers to avoid CWEs as part of their work. It's much cheaper not to have to remove issues later.
- **Typical Use:** There is commercial training for developers on how to avoid adding weaknesses to their code.
- **Typical Use:** Tool purchasers can use the list to ensure their tools look for the full range of weaknesses.

Tools to check for weaknesses Common Weaknesses

Methods	Tools/Resources
Manual design/code review	Developers trained to avoid/locate weaknesses and attacks.
Automated static analysis of code	An appropriate automated tool. Source Code. An analyst to help interpret results.
Often Combined – based on CWE	

DHS Software Assurance (SwA) Program can help your organization locate training for developers and analysts, as well as potential tools.

Lists of common weaknesses and attacks:

Common Attack Pattern Enumeration and Classification (CAPEC)

- **Source:** Academia, private sector, public sector
- **Content:** Dictionary of over 400 attacks
  - Some examples: address spoofing, brute-force encryption/password attacks, man-in-the-middle attacks, etc.
  - Typical info: attack methods/examples, common mitigation strategies, links to related weaknesses
- **Typical Use:** Dynamic Tool purchasers can use the list to ensure their tools attack the full range of weaknesses.
- **Typical Use:** Pen Testers can get guidance on how to attack known weaknesses to verify that they are not present.

Tools to check for weaknesses Attack Paths

Methods	Tools/Resources
Manual penetration testing	"White-hat" hackers trained to locate/exploit weaknesses.
Automated dynamic analysis of applications (Tools like ??)	An appropriate automated tool. Executable Code. An analyst to help interpret results.
Automated Web-testing (Tools like Web-Inspect……)	An appropriate automated tool. Access to the website. An analyst to help interpret results.
Often Combined – based on CAPEC	

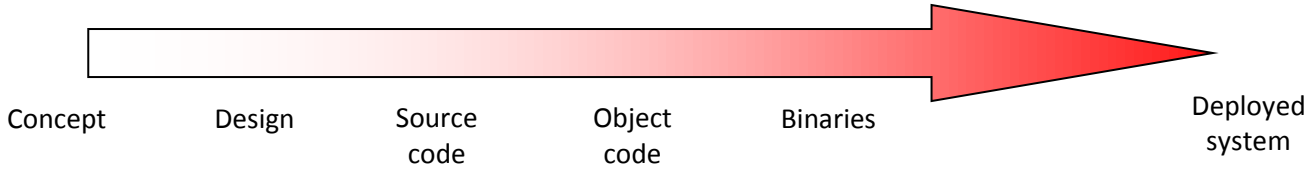
DHS Software Assurance (SwA) Program can help your organization locate training for penetration testers and analysts, as well as potential tools.

Common Weakness Scoring System (CWSS)

- **Source:** DHS - MITRE
- **Content:** Method to score (rank) weaknesses
  - Similar to the National Vulnerability Databases CVSS.
- **Typical Use:** Rank weaknesses so you can decide which ones to address first.

DHS Software Assurance (SwA) Program can help your organization understand how to use CWSS to rank weaknesses.

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

Does the system contain any of those weaknesses?

What types of attacks exploit those weaknesses?

CAPEC

Does my testing cover all of those weaknesses?

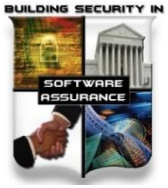
automation can help…

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

- 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**)

- Common Vulnerabilities and Exposures (**CVE**)
- Open Vulnerability Assessment Language (**OVAL**)
- Malware Attribute Enumeration and Characterization **(MAEC)**
- Cyber Obersvables eXpression **(CybOX)**

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

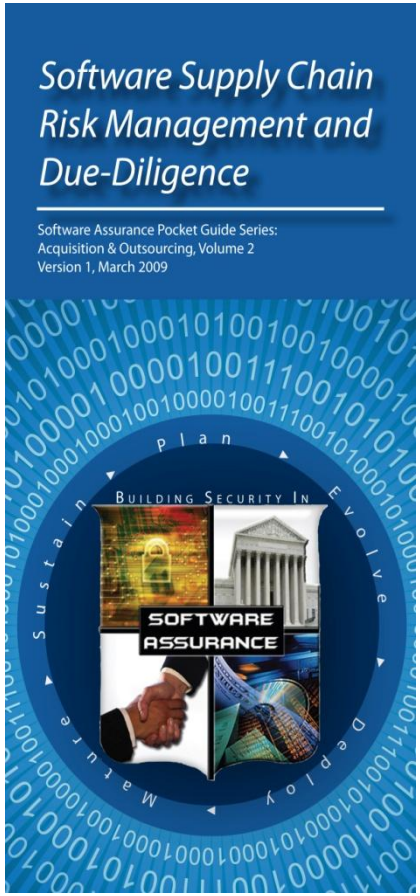
- Integrating Security into the Software Development Life Cycle
- Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses
- Risk-based Software Security Testing
- Requirements and Analysis for Secure Software
- Architecture and Design Considerations for Secure Software
- Secure Coding and Software Construction
- Security Considerations for Technologies, Methodologies & Languages

SwA Life Cycle Support

- SwA in Education, Training and Certification
- Secure Software Distribution, Deployment, and Operations
- Code Transparency & Software Labels
- Assurance Case Management
- Secure Software Environment and Assurance EcoSystem

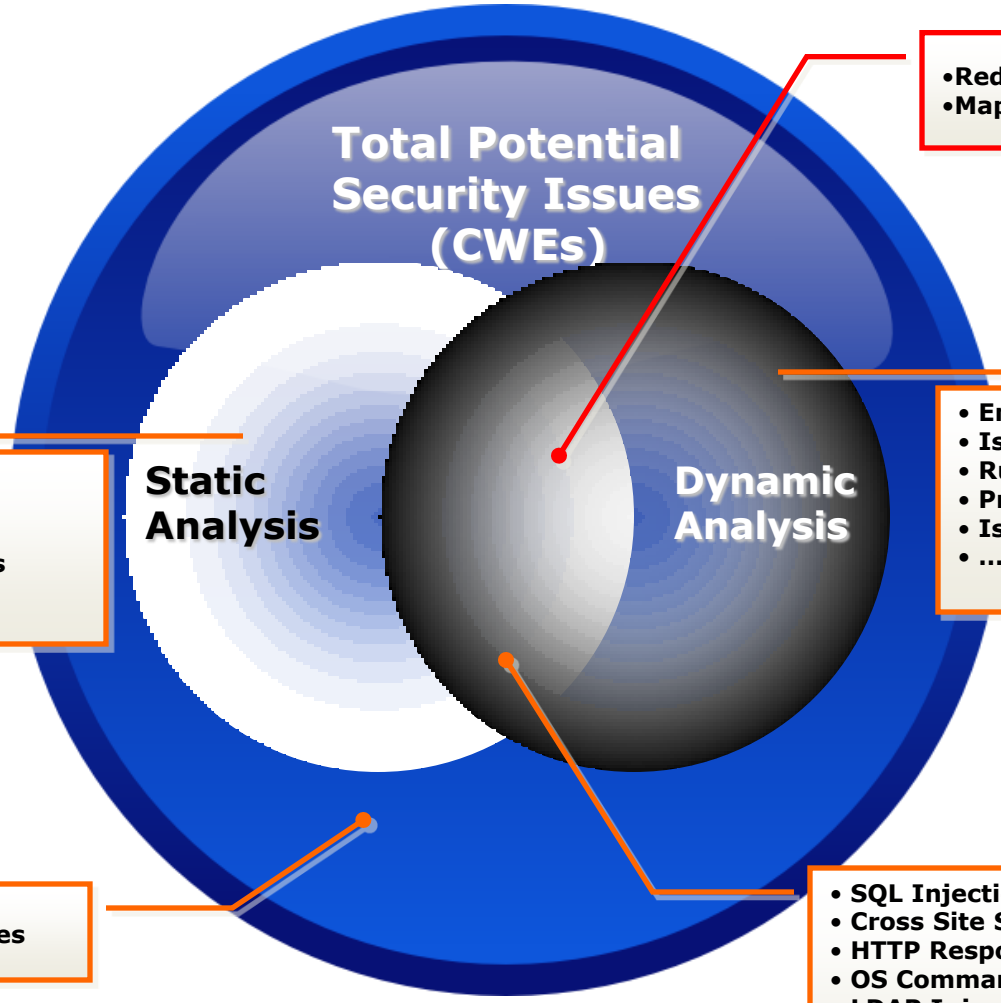
SwA Measurement and Information Needs

- Making Software Security Measurable
- Practical Measurement Framework for SwA and InfoSec
- SwA Business Case and Return on Investment



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)

Value of Aligning Multiple Perspectives



- Reduce false positives
- Map Exploited Issues to Code

- Null Pointer Dereference
- Threading Issues
- Issues in Dead Code
- Insecure Crypto Functions
- …

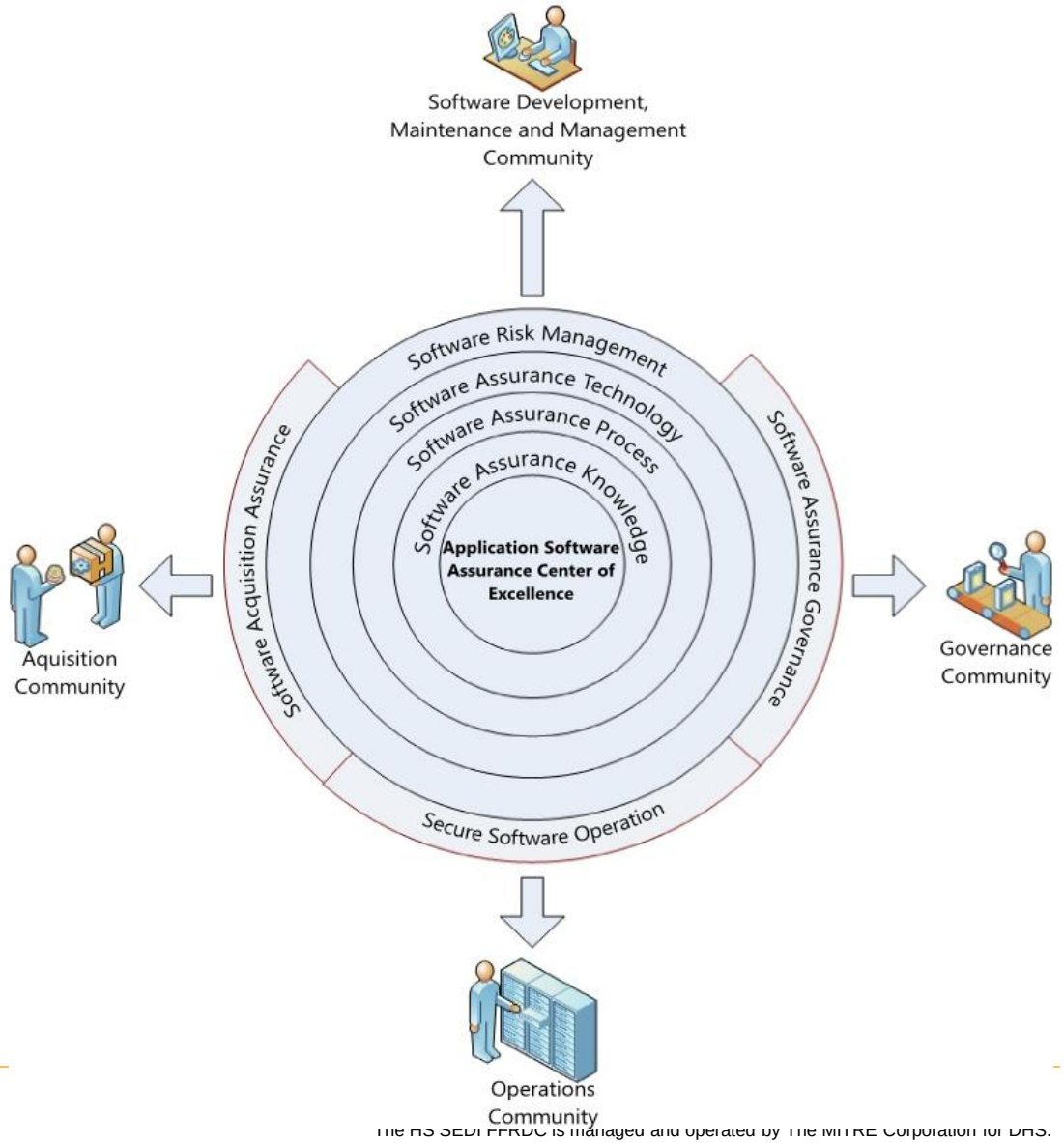
- Environment Configuration Issues
- Issues in integrations of modules
- Runtime Privileges Issues
- Protocol Parser/Serializer Issues
- Issues in 3rd party components
- …

- Application Logic Issues

- SQL Injection
- Cross Site Scripting
- HTTP Response Splitting
- OS Commanding
- LDAP Injection
- …

Practical Example: USAF ASACoE

- **Application Software Assurance Center of Excellence (ASACoE)**
- ***The Focal Point for Air Force Software Assurance (SwA) capability with the goal of reducing software-induced risk from Air Force applications.***



Overview of Triage Assessment Process

- Establish buildable source code and executable test or operational environment
- Run static source code analysis scan
- Run web application scan
- Run application data security scan
- Prioritize results analysis
- Eliminate obvious false positives
- Correlate results of different tools to confirm vulnerabilities or eliminate false positives
- Conduct remaining analysis
- Characterize and classify findings
- Create integrated findings report
- Adorn integrated report with mitigation advice for findings

ASACoE Rationale for Multi-perspective Approach

- Air Force is looking to maximize its understanding of security risk in all areas of its applications (interfaces, business logic, data tier, etc.)
- ASACoE recognizes the difficulty and complexity of analyzing application security tool scan results
- ASACoE wants to provide as much context and guidance as possible to developers for mitigation and remediation

Summary and Conclusions

- **Software Assurance analysis is increasingly becoming a high priority and is maturing in its capability**
- **Varying perspectives of analysis are available, each with their own unique value**
- **Blending multiple perspectives together yields better overall coverage and an integrated gestalt**
- **It is real and possible to begin pursuing this approach today**

Software Assurance

The level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its life cycle and that the software functions as intended. *Derived From: CNSSI-4009*

The level of **confidence that software** is **free from vulnerabilities** and **functions as intended**

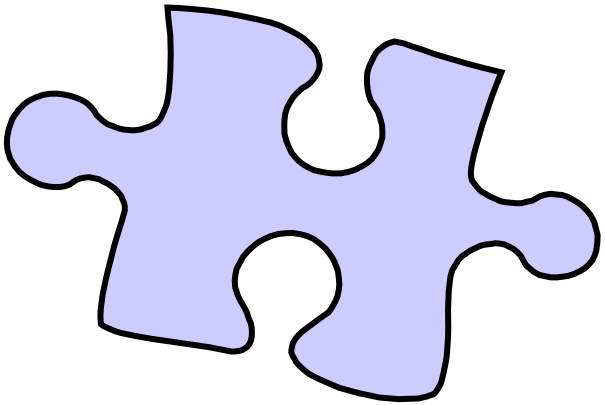
Automation

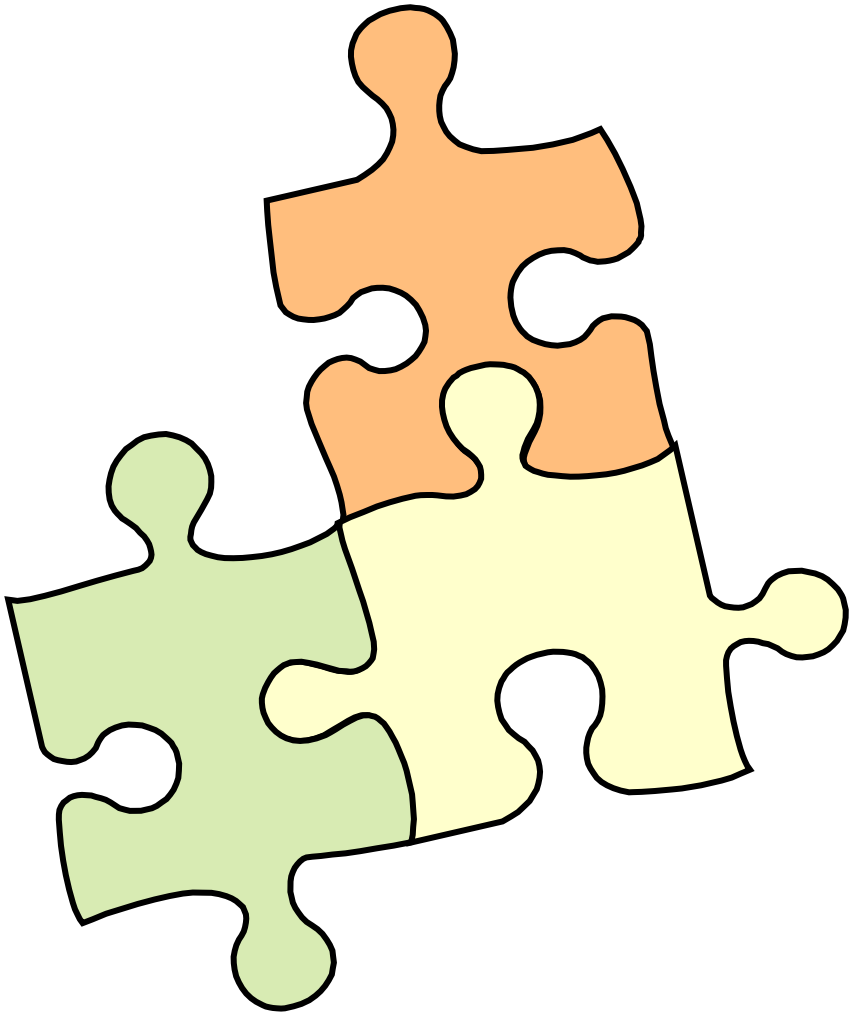
Languages, enumerations, registries, tools, and repositories

throughout the Lifecycle

Including design, coding, testing, deployment, configuration and operation

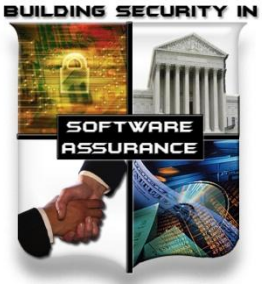
Automation is *one piece*





of the SwA puzzle.

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

Homeland Security

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 MITRE, McLean, VA – 18-20 Sep 2012

SOFTWARE ASSURANCE FORUM

"Building Security In"

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

Joe Jarzombek, PMP, CSSLP
Director for Software Assurance
National Cyber Security Division
Department of Homeland Security
Joe.Jarzombek@dhs.gov
(703) 235-3673
LinkedIn SwA Mega-Community

Homeland Security

SOFTWARE ASSURANCE FORUM

Homeland Security

Commerce

National Defense

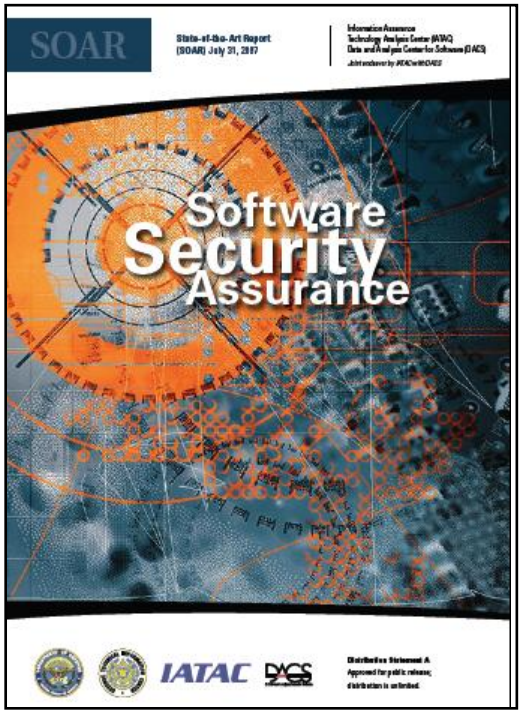
BUILDING SECURITY IN

Public/Private Collaboration Efforts for Security Automation and Software Supply Chain Risk Management

Next SwA Forum meets 18-20 Sep 2012 at MITRE, McLean, VA

State of the Art Report on Software Security Assurance

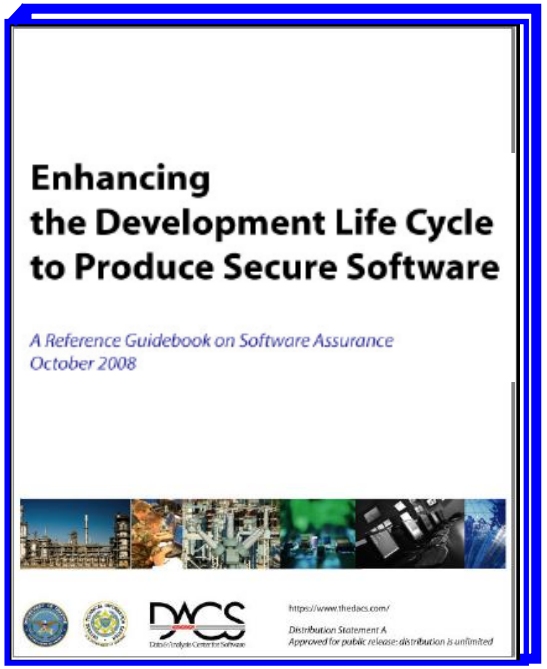
- An IATAC/DACS report identifying and describing the current state of the art in software security assurance, including trends in:
  - Techniques for the production of secure software
  - Technologies that exist or are emerging to address the software security challenge
  - Current activities and organizations in government, industry, and academia, in the U.S. and abroad, that are devoted to systematic improvement of software security
  - Research trends worldwide that might improve the state of the art for software security
- Available free via http://iac.dtic.mil/iatac/download/security.pdf



Enhancing the Development Life Cycle to Produce Secure Software

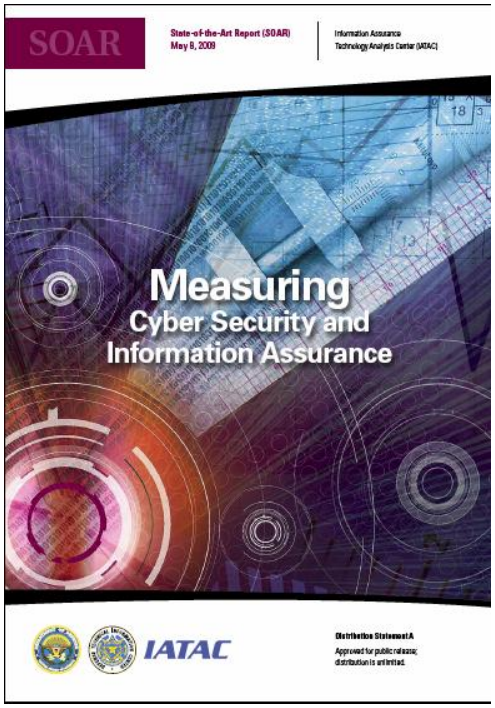
- Describes how to integrate security principles and practices in software development life cycle
- Addresses security requirements, secure design principles, secure coding, risk-based software security testing, and secure sustainment
- Provides guidance for selecting secure development methodologies, practices, and technologies
- Available free via

  https://www.thedacs.com/techs/enhanced_life_cycles/



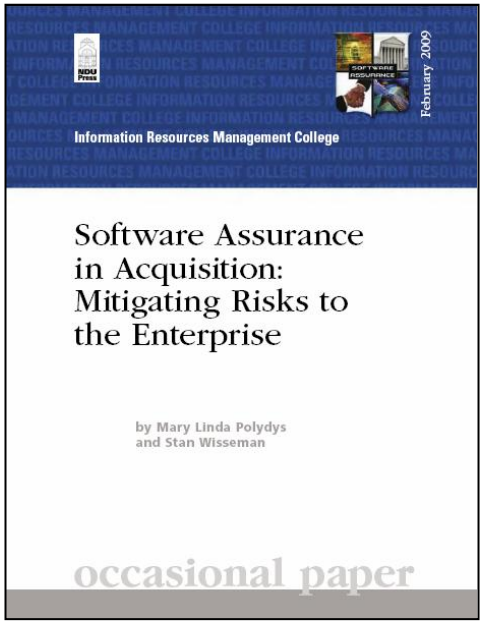
Measuring Cyber Security and Information Assurance

- Provides a broad picture of the current state of cyber security and information assurance (CS/IA), as well as, a comprehensive look at the progress made in the CS/IA measurement discipline over the last nine years since IATAC published its IA Metrics Critical Review and Technology Assessment (CR/TA) Report in 2000
- Available free via
- http://iac.dtic.mil/iatac/download/cybersecurity.pdf



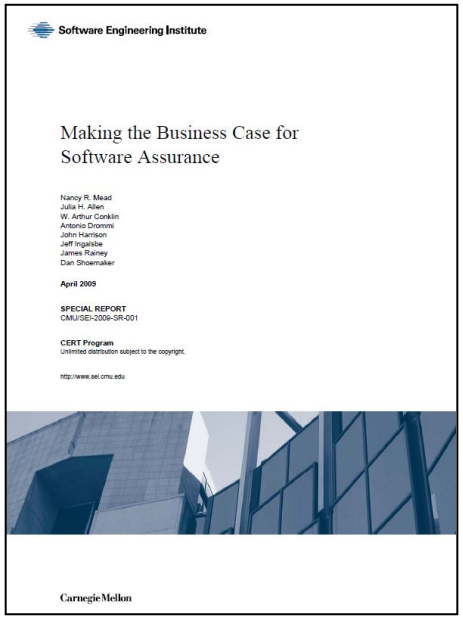
Software Assurance in Acquisition: Mitigating Risks to the Enterprise

- Provides information on how to incorporate Software Assurance considerations in key decisions
  - How to exercise due diligence throughout the acquisition process relative to potential risk exposures that could be introduced by the supply chain
  - Includes practices that enhance SwA in the purchasing process
    - Due diligence questionnaires designed to support risk mitigation efforts by eliciting information about the software supply chain (these are also provided in Word format so they can be customized)
    - Sample contract provisions
    - Sample language to include in statements of work
- Pre-publication version available free via https://buildsecurityin.us-cert.gov/swa/downloads/SwA_in_Acquisition_102208.pdf
- Final version published by National Defense University Press, Feb 2009



Making the Business Case for Software Assurance

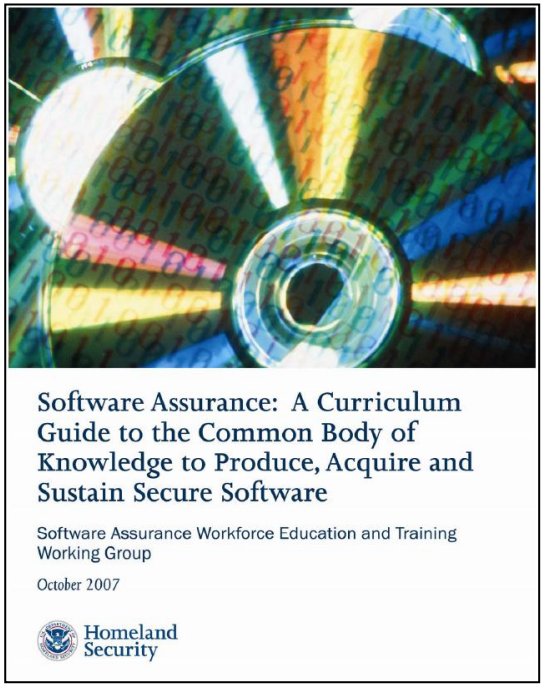
- Provides background, context and examples for making the business case for software assurance:
  - Motivators
  - Cost/Benefit Models Overview
  - Measurement
  - Risk
  - Prioritization
  - Process Improvement & Secure Software
  - Globalization
  - Organizational Development
  - Case Studies and Examples
- Available free via
- http://www.sei.cmu.edu/library/abstracts/reports/09sr001.cfm



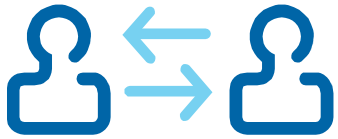
Software Assurance: A Curriculum Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software

- Provides a framework intended to identify workforce needs for competencies, leverage sound practices, and guide curriculum development for education and training relevant to software assurance
- Available free via

  https://buildsecurityin.us-cert.gov/bsi/940-BSI/version/default/part/AttachmentData/data/CurriculumGuideToTheCBK.pdf



Useful Links



- DHS Build Security In Web Site
  - A wealth of software and information assurance information, including white papers on static code analysis tools
  - More information on Build Security In can be found at: https://buildsecurityin.us-cert.gov/daisy/bsi/home.html
- Common Weaknesses Enumeration (CWE)
  - This site provides a formal list of software weakness types created to Serve as a common language for describing software security weaknesses in architecture, design, or code
  - More information on CWEs can be found at:
    - http://cwe.mitre.org/
- CWE/SANS Top 25 Most Dangerous Software Errors
  - The 2010 CWE/SANS Top 25 Most Dangerous Software Errors is a list of the most widespread and critical programming errors that can lead to serious software vulnerabilities.
  - More information on the CWE/SANS Top 25 can be found at:
    - http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.pdf
- NIST SAMATE Static Analysis Tool Survey
  - The National Institutes for Science and Technology (NIST), Software Assurance Metrics and Tool Evaluation (SAMATE) project, provides tables describing current static code analysis tools for source, byte, and binary code analysis
  - More information on SAMATE can be found at: http://samate.nist.gov/

Working for Homeland Security

The DHS Office of Cybersecurity and Communications (CS&C) serves as the national focal point for securing cyber space and the nation's cyber assets.

CS&C is actively seeking top notch talent in several areas including:

- Software assurance
- Information technology
- Telecommunications
- Program management
- Public affairs

To learn more about CS&C and potential career opportunities, please visit USAJOBS at www.usajobs.gov .

Homeland Security