Incorporating Software Assurance into Department of Defense Acquisition Contracts

Working Paper, November 15, 2017

Department of Defense (DoD) Software Assurance (SwA)
Community of Practice (CoP) Contract Language Working Group

Working Group Chair: John R. Marien
Co-Chair: Robert A. Martin

Office of the Deputy Assistant Secretary of Defense for
Systems Engineering

Washington, D.C.

Department of Defense (DoD) Software Assurance (SwA) Community of Practice (CoP) Contract Language Working Group. *Incorporating Software Assurance into Department of Defense Acquisition Contracts.* Working paper. Washington, D.C.: Office of the Deputy Assistant Secretary of Defense for Systems Engineering, November 15, 2017.

Deputy Assistant Secretary of Defense
Systems Engineering
3030 Defense Pentagon
3C167
Washington, DC 20301-3030
osd.atl.asd-re.se@mail.mil
www.acq.osd.mil/se

Contents

Figures

Tables

Preface

This document provides a discussion of ways Department of Defense (DoD) acquisition efforts, both new and existing, may incorporate language regarding software assurance (SwA) concepts and deliverables into new or existing contracts. The goal is to ensure delivery of software to the operators that behaves securely and is resilient to hazards and attack while fulfilling the mission that the software supports.

All programs use software to some extent. The examples here are not intended as copy-and-paste contract language but rather as samples that programs may use and tailor for their particular contracts. Ideally, SwA language should be included in the Request for Proposals (RFP); however, as many programs are already under contract, this document also addresses how and where to apply SwA contract language throughout the software development life cycle.

The introduction in Section 1 discusses the definition and interpretation of software assurance. Section 2 highlights the locations in the RFP and contract where programs should consider inserting software assurance-related language. Section 3 discusses automated software vulnerability detection tools. Section 4 focuses on language relating to deliverables, and Section 5 provides examples of incentives and award structures.

The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)) sponsors this effort, which is organized through the DoD SwA Community of Practice (CoP) Contract Language Working Group. ODASD(SE) published the initial working paper in February 2016. This 2017 version addresses comments from the office of Defense Procurement and Acquisition Policy, provides an updated sample Data Item Description (DID), and includes updates to the reference information. Further revisions are forthcoming to incorporate new insights, feedback, from practitioners, and to address ideas that have been developed in the DoD, other parts of the government, and industry.

1 Introduction

Common industry practice and Section 933 of the National Defense Authorization Act for Fiscal Year 2013 (Public Law 112-239) [1] define software assurance (SwA) as

> 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.

The DoD *Program Protection Plan (PPP) Outline and Guidance* [2] Software Assurance Table [2, 3] and the Defense Acquisition Guidebook (DAG) Chapter 9, Program Protection [4], describe measures and approaches defense acquisition programs can use to conduct SwA for a system.

This document suggests language that programs might tailor for use in RFP packages and contracts to provide the program office with insight into software development activities of its contractors and to provide assurance regarding the developed software's ability to meet mission needs. With so much of today's mission functionality realized through software—both traditional software used for planning, management, and logistics as well as software used directly in weapon systems, vehicles, infrastructure management, and utilities—the need for assurance about that software continues to grow.

The applicable language generally will appear in Sections C, L, and M of the standard format RFP and contract. When there are Federal Acquisition Regulation (FAR) or Defense Federal Acquisition Regulation Supplement (DFARS) clauses related to SwA, they will appear in Section I.

The following subsections examine each part of the definition of software assurance: *confidence* in a system, that the system *functions as intended*, and that the system is *free of vulnerabilities*. The succeeding sections further discuss how applicable language can be used in an RFP and contract to address the three key concepts of the SwA definition. This document addresses only detectable vulnerabilities, as undetectable vulnerabilities could not be used by an adversary.

1.1 Confidence

Confidence regarding contractors' assurance activities comes from obtaining appropriate information that a program office and others can understand and that supports the claims about the functionality of the software as well as the addressing of exploitable constructs in the system. The use of standardized collections of weaknesses, vulnerabilities, and attack patterns makes the understanding of contractors' assurance actions easier and more consistent and offers opportunities for reuse of that approach to provide similar confidence in other systems needed and in other contractors. There are several points in the life cycle of a system when insights into the risks and the mitigation of those risks can be obtained and the PPP Software Assurance Table calls out several of them explicitly, as shown in Figure 1, so that the appropriate information can be collected and reviewed at the earliest stage of development supporting risk management decision making.

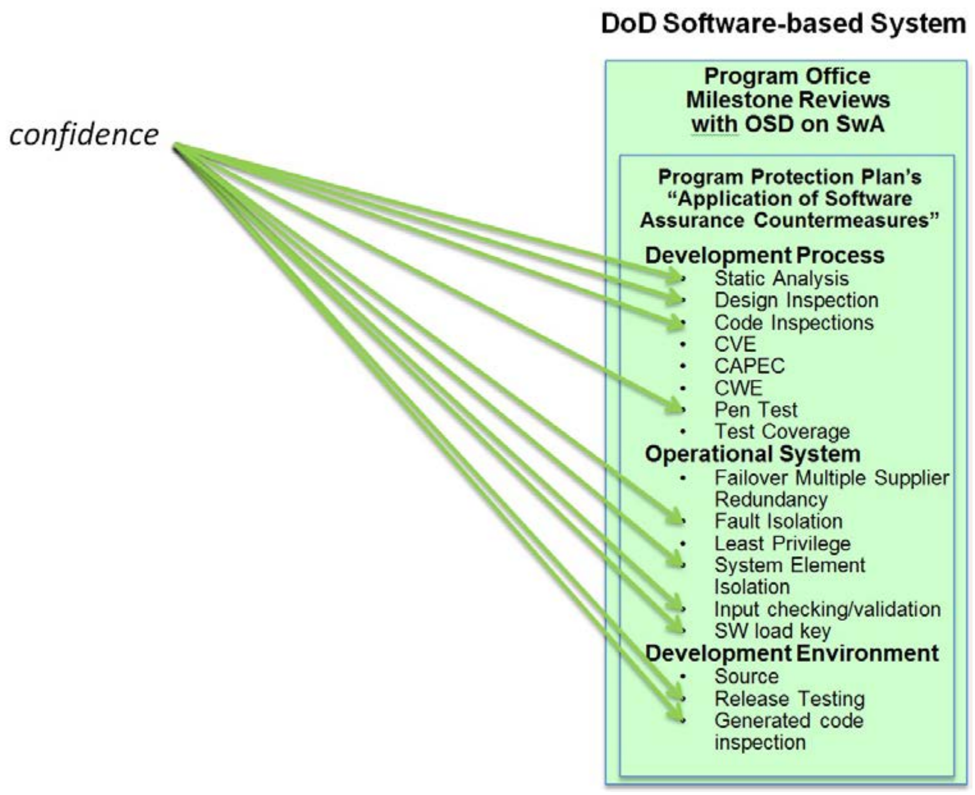

DoD Software-based System
confidence
Program Office Milestone Reviews with OSD on SwA
Program Protection Plan's "Application of Software Assurance Countermeasures"
Development Process
Static Analysis
Design Inspection
Code Inspections
CVE
CAPEC
CWE
Pen Test
Test Coverage
Operational System
Failover Multiple Supplier Redundancy
Fault Isolation
Least Privilege
System Element Isolation
Input checking/validation
SW load key
Development Environment
Source
Release Testing
Generated code inspection

Figure 1. Confidence

We build confidence in the software system through *static analysis*, *design inspection*, *code inspections*, and *penetration testing* during the development process. Each of these activities identifies weaknesses and vulnerabilities in the software and its design and allows easy, and early, correction at minimal cost and time.

1.2 Functions as Intended

Determining whether a system "functions as intended" requires both showing through testing that the intended functionality is there and through test coverage and metrics understanding the system does not perform unrequired functions. As with the confidence measures discussed above, there are several points where insights into the risks and the mitigation of those risks regarding a system's ability to "function as intended" can be obtained. The PPP Software Assurance Table calls out several of them explicitly, as shown in Figure 2, so that the appropriate information can be planned for, collected, and reviewed at the appropriate stage of the System Development Life Cycle (SDLC). The Common Attack Pattern Enumeration and Classification (CAPEC) [5, 6] contains a collection of patterns of attacks that can be used to describe misuse and abuse testing done on a system either through automated tools or through pen test team activities.

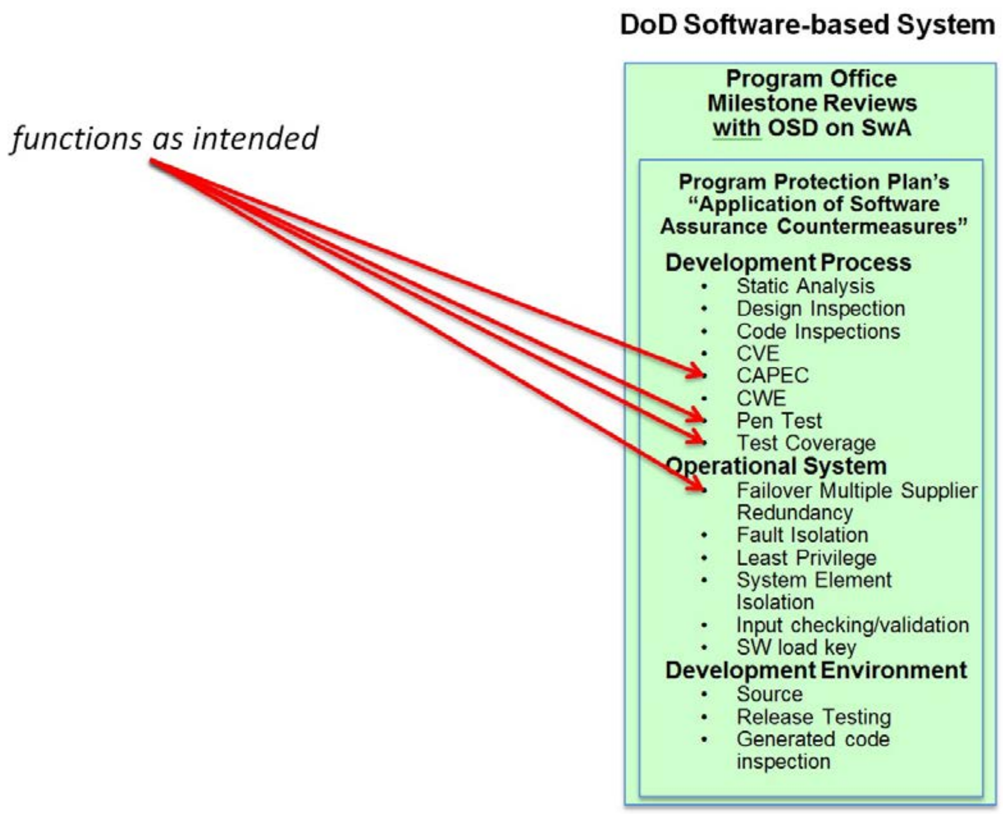


Figure 2. Functions as Intended

We assert that the software *functions as intended*, and only as intended, through application of *CAPECs*, *penetration testing*, *test coverage*, and through *the use of multiple-redundant implementations of critical elements by different suppliers and having the system failover to an alternate implementation.* Each of these activities allows us to identify that the software is indeed functioning as intended without functioning, or being made to function, in unintended ways.

1.3 Free of Vulnerabilities

It is common practice in industry [7, 8, 9, 10, 11, 12, 13, 14, 15] to refer to the Common Weakness Enumeration (CWE) [16, 17, 18] catalogue when assessing whether software is "free of vulnerabilities." This approach allows others to understand both what was "looked for" and what was not but could have been "looked for."

Similarly, for commercial software packages (proprietary and open source) used as part of a system, the collection of publicly known vulnerabilities in these types of software, called the Common Vulnerabilities and Exposures (CVE) [19, 21] dictionary, is almost always used as a reference source to determine if known issues have been mitigated.

We assert that the software is *free of vulnerabilities* by validating that those CVE and CWE items that are most dangerous to the mission are absent from the software and that the software operates at the *least privilege* required to complete its task.

Figure 3 shows how the listed types of information can give insights into activities and their results with respect to determining whether a system is “Free of Vulnerabilities” as outlined in the PPP Software Assurance Table and explained in DAG Chapter 9. Of course, obtaining these findings and informational items for a particular system during its design, development, test, and post-deployment sustainment activities will require that the contract provide for the contractor to collect the necessary information or provide the Government with the opportunity to collect it directly.

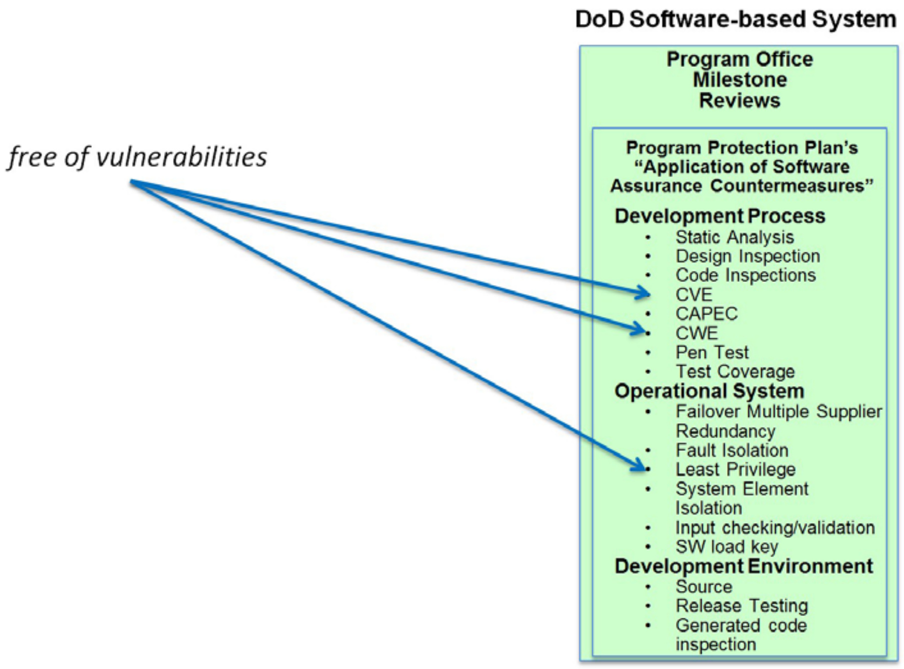


Figure 3. Free of Vulnerabilities

Other parts of a contract will guide or require these different artifacts and activities from a contractor, and it is critical that the contract fully describe and specify what is expected from the contractor’s development activity. The System Performance Specification, the Statement of Work, Systems Engineering Plan, Incentive Plan, Contract Data Requirements Lists (CDRL) and other parts of a contract all have their appropriate part in expressing the expectations and performance of the developing organization and ensuring that they and their DoD customer meet each other’s expectations and concerns; however, these activities are all predicated on the Government having access to the custom and reused software for independent inspection, testing, and evaluation, so including a paragraph about delivery of the source code, all libraries, and frameworks is crucial.

A notional Statement of Objectives (SOO), as shown in Figure 4, could cover software assurance-related activities for the software development, the software interfaces, and the use of software tools and metrics.

Software Development:

It is the Government's objective for the contractor to develop secure, reliable, resilient, assured software:

- The contractor shall develop a secure coding guide which specifies language, required constructs/practices/patterns, prohibited constructs/practices/patterns, software comment requirements
- The contractor shall implement formal (e.g., Fagan) design and code inspections
- The contractor shall verify all code against the CWE, CVE, OWASP
- The contractor shall perform an origin analysis of all third-party libraries and frameworks used and ensure the versions used in the delivered software have no publicly known vulnerabilities and continue to review for newly reported vulnerabilities throughout the sustainment of the delivered software
- The contractor shall ensure all developers are trained and held accountable for development of secure code

Software Interfaces:

It is **the** Governments objective for all software interfaces to be completely documented and utilize strong typing and range checking

Software Tools:

It is **the** Governments objective for the contractor to utilize automated tools to maximize efficiency and effectiveness:

- The contractor shall utilize automated tools to support software configuration control
- The contractor shall utilize automated testing tools to support regression testing for all custom code with at least 90% statement coverage.
- The contractor shall utilize at least 2 different commercial static analysis tools to measure software quality and access vulnerabilities. Multiple tools are to be used to improve detection of software quality issues and vulnerabilities as each tool uses different detection methods.

Software Metrics:

<**etc**.>

Figure 4. Notional SOO Items

Working from the System Description document, we can map these concepts to typical contract sections as illustrated in Figure 5.

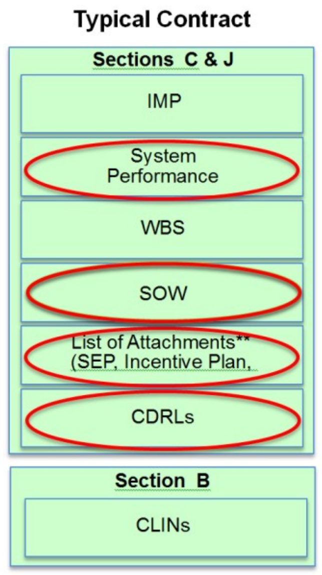


Figure 5. Typical Contract Structure

Using the SOO, we can identify notional SOO items related to SwA. The specific wording for each SOO item should be adjusted as required for a specific contract.

2 Inserting Software Assurance into Contract Language

Figures 6 through 9 illustrate that if SwA activities are to be followed throughout the life cycle, they must be inserted into the appropriate contract points through the contracting process. SwA must be applied throughout the software life cycle and therefore must be addressed in contract vehicles from the RFP through sustainment.

If a program has an existing contract in development or in sustainment and needs to add SwA items, the changes must be inserted through a bilateral contract modification agreed to by the contractor, as illustrated in Figure 6. Adding these types of risk reduction activities and measurement opportunities to an existing contract can be disruptive and costly. The risk to the operational system posed by the unknown and unexpected vulnerabilities and frailties must be considered against the potential cost and perturbation to the project of identifying and removing them.

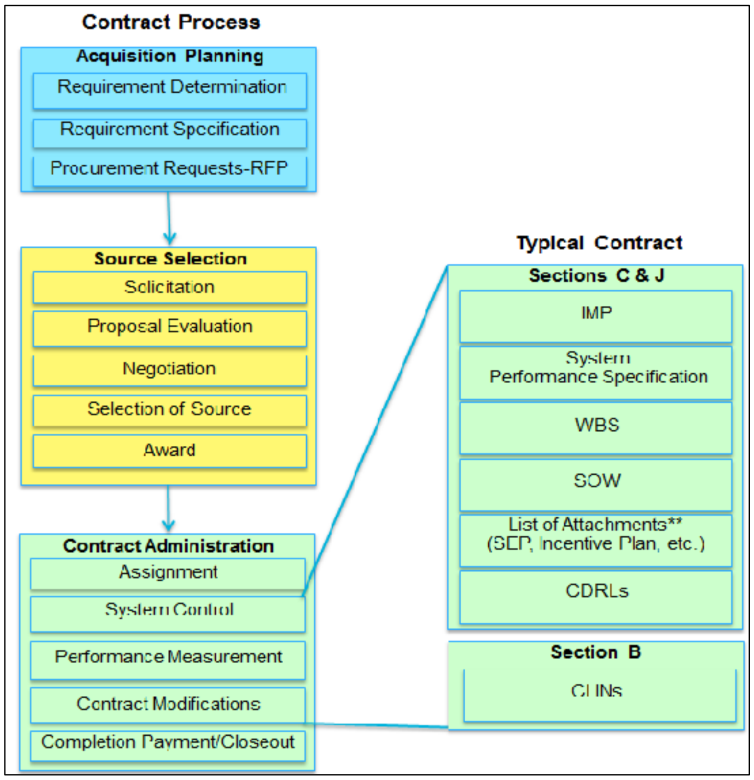


Figure 6. Contract Process for Existing Contract in Development or in Sustainment

Whether a program has an existing contract or is in the process of establishing one as shown in Figures 7 through 9, the allocation of liability for software defects and vulnerabilities must either be put in the contract directly through the mechanisms described in the subsequent parts of this document and the referenced examples, or the program can try to rely on normal liability protections from either the Uniform Commercial Code (UCC) if there is a case that the software in question is "goods" versus a "service" and the developer may be subject to suit under a strict liability theory of tort for residual vulnerabilities and susceptibility to hazards and attacks.

The program office's concerns about liability have to be explicitly described, and must be discussed so both the developer and the Government have a clear understanding, in advance, of what is and is not the result when flaws are found in the operational system and fixes are needed or the impact from a failure needs to be addressed.

For those interested in understanding the general liability issues surrounding software, two good starting points on the question of contractor liability for software issues is the May 26, 1990, article posted at the *Berkeley Technology Law Journal* (BTLJ) Volume 5 Issue 1, "Software Product Liability: Understanding and Minimizing the Risks" [21] by Lawrence B. Levy and Suzanne Y. Bel, and the article by Michael D. Scott, "Tort Liability for Vendors of Insecure Software: Has the Time Finally Come?" [22].

The question of whether the development team is liable for fixing the residual issues in their software, or if they hold liability for the damage or fall-out from the vulnerable and/or unreliable nature of the software allowed to happen or caused, is still open. Until such time as penalties are levied against software vendors and developers who release faulty code, there will be insufficient incentive for software vendors and developers to build security into their software. The only current alternative is to explicitly state what faults are to be repaired by the software vendor/developer in the contract agreed to by both parties.

In a new procurement, the program office will need to make sure the appropriate parts of the contract include the right language. To do so the program will have to make sure that the Source Selection process as shown in Figure 7 results in a contract with the needed language in the right parts. To do that, the appropriate language needs to be part of the selected proposal or proposals, as shown in Figure 8, which means it needs to be included in the RFP and addressed by the criteria used to evaluate those proposals, as shown in Figure 9.

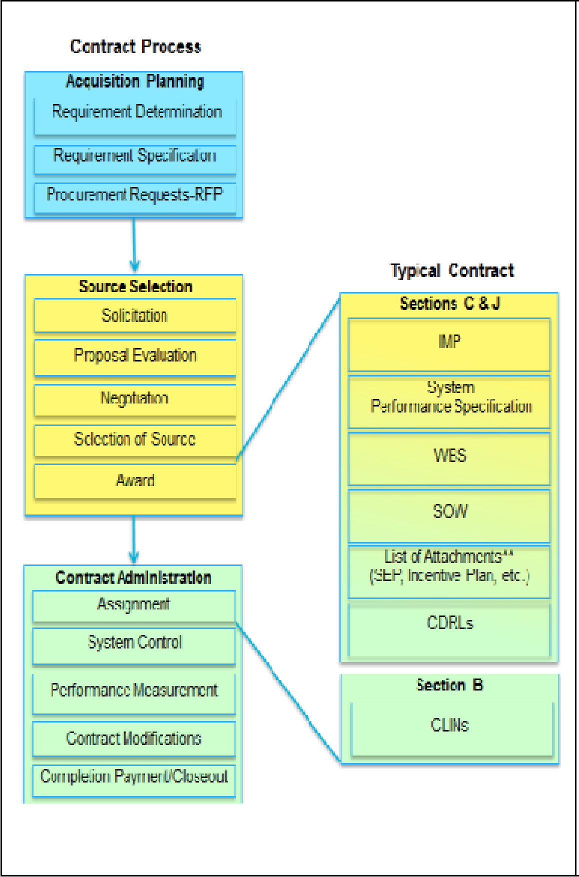


Figure 7. Contract Process

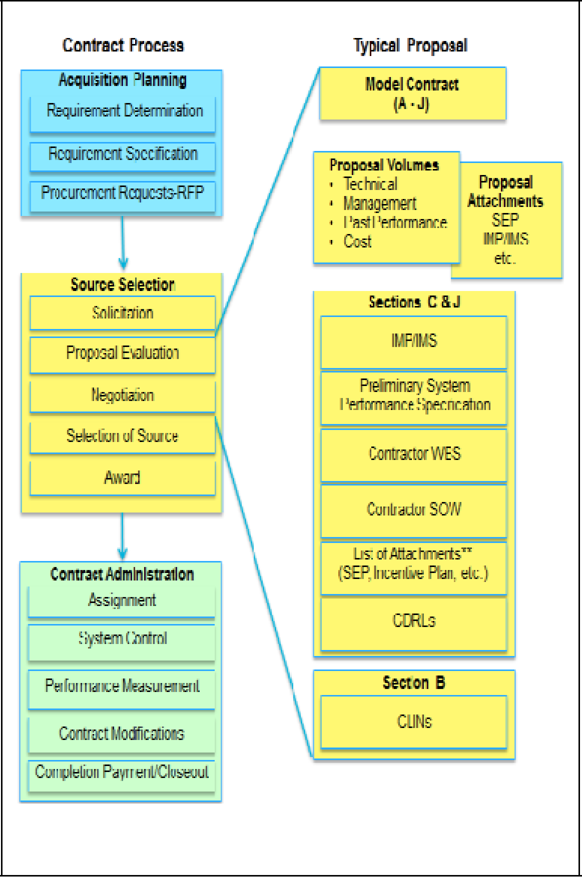


Figure 8. Contract Process During Proposal

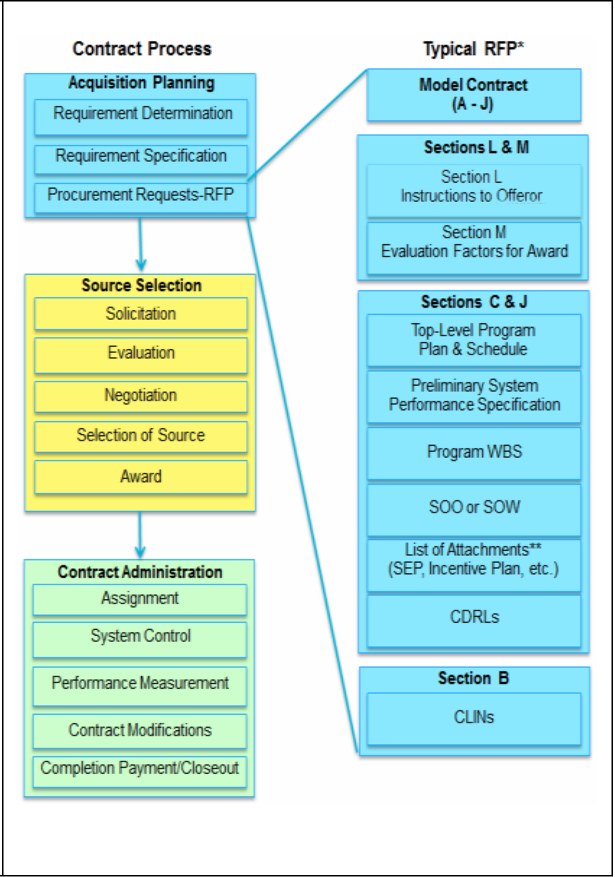
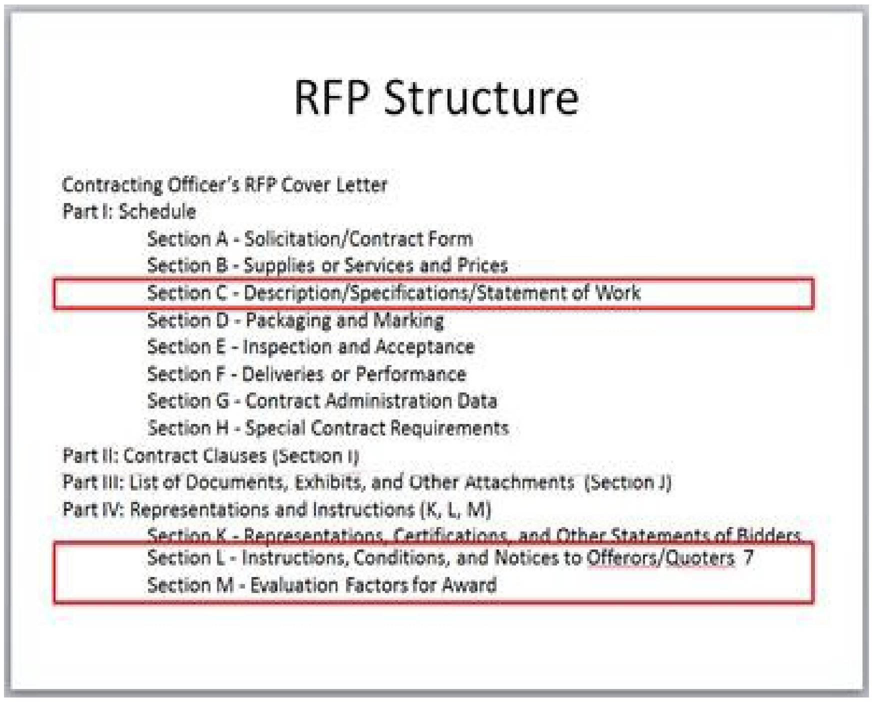


Figure 9. Contract Process RFP

For a new procurement to incorporate the appropriate software assurance activities into the contract there must be a clear articulation by the Government about what is required.

In addition, the selection criteria in the final award, the source selection process, and the performance criteria and incentive plan for the awarded contract all must include a clear statement of the requirements for software quality and verification thereof.

Within the traditional RFP package, the appropriate language to be inserted goes in Section C, and Sections L and M, as highlighted in Figure 10.



RFP Structure

Contracting Officer's RFP Cover Letter
Part I: Schedule
- Section A - Solicitation/Contract Form
- Section B - Supplies or Services and Prices
- Section C - Description/Specifications/Statement of Work
- Section D - Packaging and Marking
- Section E - Inspection and Acceptance
- Section F - Deliveries or Performance
- Section G - Contract Administration Data
- Section H - Special Contract Requirements

Part II: Contract Clauses (Section I)
Part III: List of Documents, Exhibits, and Other Attachments (Section J)
Part IV: Representations and Instructions (K, L, M)
- Section K - Representations, Certifications, and Other Statements of Bidders
- Section L - Instructions, Conditions, and Notices to Offerors/Quoters 7
- Section M - Evaluation Factors for Award

Figure 10. RFP Structure

The subsequent examples and other procurement language examples target the above-mentioned RFP Section by name. For example, the Air Force Life Cycle Management Center (AFLCMC), AFLCMC/EZC – "Engineering Model RFP" (EM RFP) [23] language package provides modules with suggested items for several sections of an RFP to address different issue areas. These areas include modules on Software Assurance, Software Engineering, Software Metrics, and Supply Chain Risk Management (SCRM), Independent Verification and Validation, Quality Assurance, System Security Engineering, Configuration Management, Anti-Tamper, Firmware, and Internal and External Interfaces.

Some of the examples do not offer specific language but discuss the issues that need to be addressed, while others are examples of existing contract language in use by DoD Components or other Government agencies to address some SwA issues.

The important point to remember is that if a program wants a contractor to follow a particular approach or report and discuss issues dealing with software assurance in a particular manner, the program must specify that approach or report and require the discussion of those items in the contract, deliverables, and incentive plan. DoD must show to the contractor that the Government values quality and assurance by explicitly tying the delivery of quality software and software free of vulnerabilities to financial rewards and delivery criteria.

We must not sacrifice critical aspects of quality for reduced cost or improved schedule when those diluted requirements put the operational integrity of the software in question. In order for software to support the mission(s) it is planned to support, we must give the contractor good technical requirements (e.g., develop resilient design), require good practices (e.g., use code inspections), structure deliverables to require reporting issues in a standards-based manner (with CVEs, CWEs, and CAPECs), and support quality both initially at the time of delivery and in operation and sustainment.

2.1 Section I

Section I contains clauses defining the rights and responsibilities of the contracting parties and clauses required by procurement regulations or laws that pertain to the procurement.[1]

2.2 Section C Example

A sample Section C would include instructions to the offeror to require that all SwA requirements applied to the Prime contractor also be applied by the Prime contractor to all subcontractors, or flow-down. Example entries for section C could be:

Section C:

- The software assurance requirements detailed in this RFP shall apply to all of the software delivered by the Prime contractor. Should the Prime contractor employ subcontractors, then the Prime contractor shall require all of the software assurance requirements detailed in this RFP from each subcontractor they employ for use on this contract.
- All software assurance requirements defined herein shall apply to all reused code included and delivered by the Prime contractor and its subcontractors.
- All software assurance requirements defined herein shall apply to all reused objects, both proprietary and open source, and their originating reused source code, whether that code was included and delivered by the Prime contractor and its subcontractors or just referred to by them as the source of the reused object.
- The Government will provide to the Prime contractor a list of the Top-*n* most important CWEs. All software delivered by the Prime contractor shall be free of all of the CWEs in the Government's Top-*n* CWE list. If a CWE in the Government's Top-*n* CWE list is found to be

[1] *Department of Defense COR Handbook.* Director, Defense Procurement and Acquisition Policy, OUSD(AT&L), March 22, 2012. http://www.acq.osd.mil/dpap/cpic/cp/docs/USA001390-12_DoD_COR_Handbook_Signed.pdf

present in the delivered software, the Prime contractor shall be liable only to repair the defect within XX-days at the contractor's expense.

The Joint Federated Assurance Center (JFAC) SwA community is developing and applying innovative approaches to incorporating requirements in their RFPs. The successor to this document will report both requirements and strategies that have proven workable, and those to avoid.

2.3 Sections L and M Examples

It is important that for each instruction, condition, and notice to offerors included in Section L of an RFP that there be a corresponding paragraph in Section M indicating how to evaluate the item from Section L. An example entry for section L is:

Section L: Paragraph XX: Software Assurance Sample Report:

- The Offeror shall provide a report titled "Software Assurance Analysis."
- The "Software Assurance Analysis" report shall contain two sections:
    - Section 1: Details how static analysis for Software Assurance is used within the offeror's development life cycle in writing and in diagrammatic form for illustrative purposes.
    - Section 2: Contained within a spreadsheet file, details the output from a minimum of two software static analysis tools used by the offeror according to the following table (not including the two samples provided):

CWE #	Severity or Grade	Violation Description	Violation Location	Tool Name	Notes (Optional)
CWE-89	1	Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)	C:\Sample\Project\XYZ ZY.cs [line:23]		
CWE-481	2	Assigning instead of Comparing	C:\Sample\Project\XYZ ZY.cs [line:203]		

Figure 11. Sample Output of Static Analysis Tools Used

- The "Software Assurance Analysis" report shall provide a static analysis of the legacy (Government-furnished information (GFI)) code provided.
  [Make this section as specific as possible to include file names or code groups where possible.]
- The Software Assurance Analysis report shall have a primary sort on Column 1 so that all **CWE-###** entries are at the top of the report in ascending order by CWE #
- The Software Assurance Analysis report shall have a secondary sort on Column 2 so that the highest priority results are at the top of the report

Section M: Paragraph XX: The Government has run a static analysis tool on the specified code to serve as a baseline to spot-check known violations in the sample source code provided by the

Government as part of the bidder's library. The Software Assurance report will be graded as follows (100 points total + bonus points):

- Report named correctly (10 points)
- Report formatted correctly (10 points)
- Report covers the specified source code from **Section L paragraph 23** and provides text describing detailing how, and in which part(s) of the software development life cycle, static analysis for software assurance is used (10 points)
  [This Section M item corresponds exactly to the specified Section L item.]
- Report entries include CWE numbers (60 points)
  - 1-5 unique CWE entries (10 points)
  - 6-10 unique CWE entries (20 points)
  - 11+ unique CWE entries (30 points)
- Report entries include non-CWE labeled violations (10 points)
- Bonus: 10 points if each of the following 10 CWEs are identified in the sample source code. Note that there is no guarantee that the following CWEs are present in the sample source code:
  [This is a good place to insert the program's Top-N CWE list]

  1. CWE-xxx
  2. CWE-xxx
  3. CWE-xxx
  4. CWE-xxx
  5. CWE-xxx
  6. CWE-xxx
  7. CWE-xxx
  8. CWE-xxx
  9. CWE-xxx
  10. CWE-xxx

The challenge of gaining assurance about the software is not unique to DoD contracting. Many have attempted to bring the risk from insecure and brittle software under control and these previous efforts have covered some portion of what is now known as software assurance. We have reviewed many of these previous works [25, 26, 27, 28, 29, 30, 31, 32, 33], as shown in Figure 11, and believe that many of them created content that could be incorporated within today's software assurance efforts as appropriate material for programs to consider when looking for ideas on managing these risks, rather than starting from scratch.

C- Description	J – Top-Level Schedule	J – Prelim. Performance Spec.	J – Program WBS	J – SOO/SOW	J – SEP	J – CDRLS	J – List of Attachments	L – Instruction to Offerors	M – Evaluation Factors	**RFP Coverage**
				X				X		[25] Suggested Language to Incorporate System Security Engineering for Trusted Systems and Networks into Department of Defense Requests for Proposals
				X						[26] Software Assurance in Acquisition and Contract Language
		X			X	X	X	X	X	[27] Guide for Integrating Systems Engineering into DoD Acquisition Contracts
		X		X				X	X	[28] DoD Open Systems Architecture Contract Guidebook
		X		X				X	X	[29] Software Assurance in Acquisition: Mitigating Risks to the Enterprise
X	X		X	X	X	X	X	X	X	[30] Recommended Software Assurance Acquisition Language for Space & Missile Systems Center
				X						[31] Cyber Security Language for Information Technology (IT) Requirements
				X						[32] DoD/VA integrated Electronic Health Record (iEHR) Technical Specifications Summary
		X		X						[33] OWASP Secure Software Contract Annex

Figure 12. Guidance Coverage Comparisons

3 Use of Automated Detection Software Tools

Identifying most CWEs by hand, or visual inspection, is difficult, time-consuming, and error-prone. Automated tools are needed to effectively find these types of issues in software. DoD contracts should say this explicitly, but the tools must be supplemented with other types of assessment as well, such as: attack surface analysis, a security review of the CONOPs for the application, an architecture review, a design review, fuzz testing, running misuse and abuse test cases, dynamic analysis and web testing, and pen testing, as well as blue and red team assessment testing of the live application. The goal is to document the weaknesses targeted by these activities or the attack patterns emulated/simulated/practiced, and to account for the coverage/completeness of the weakness/attack spaces being addressed [34, 35].

By understanding the coverage of various types of assessment, as shown in Figure 12, a combination can be constructed to cover the CWEs of most concern, i.e., the Top-N CWEs that are important to you and the mission the application is supporting, as illustrated in Figure 13.

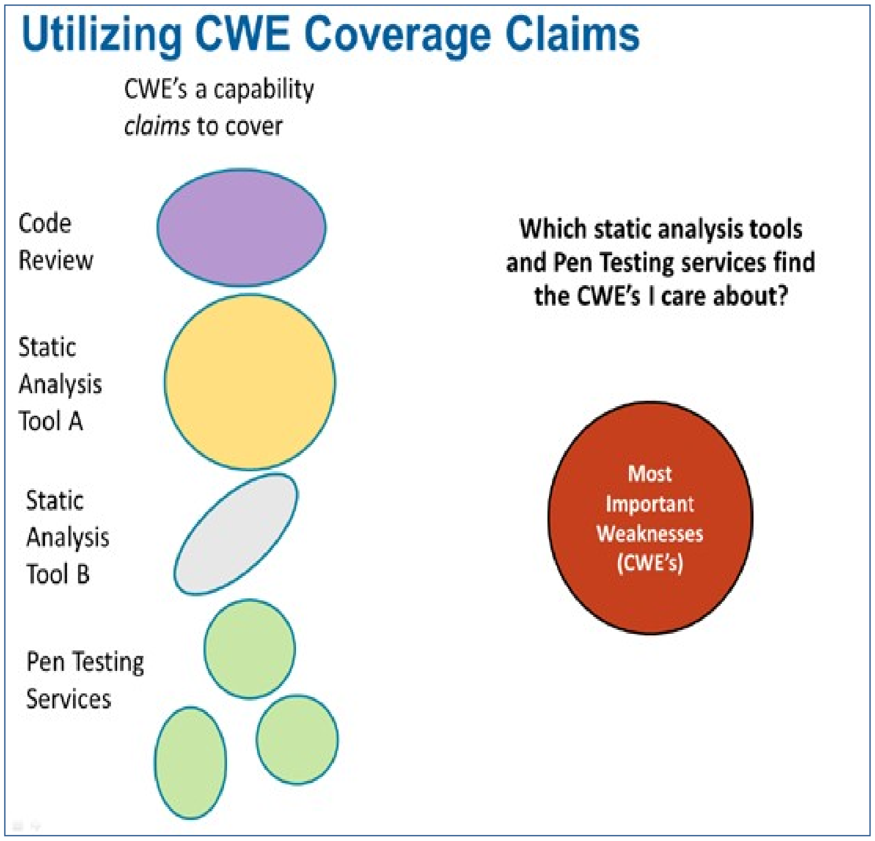

Utilizing CWE Coverage Claims
CWE's a capability claims to cover
Code Review
Static Analysis Tool A
Static Analysis Tool B
Pen Testing Services
Which static analysis tools and Pen Testing services find the CWE's I care about?
Most Important Weaknesses (CWE's)

Figure 13. Using CWE Coverage Claims

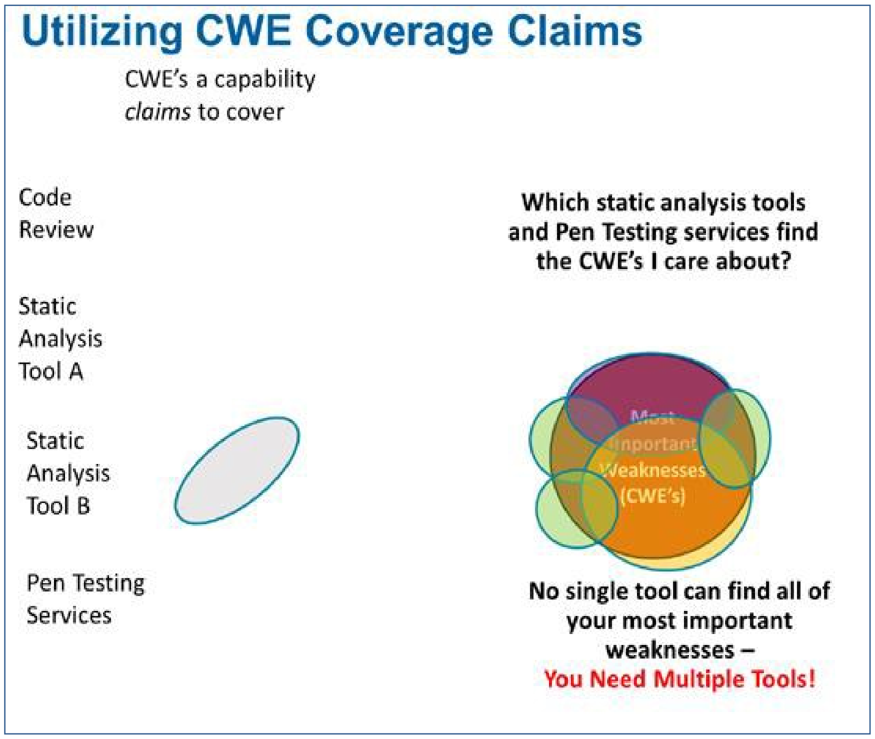

Utilizing CWE Coverage Claims
CWE's a capability *claims* to cover
Code Review
Static Analysis Tool A
Static Analysis Tool B
Pen Testing Services
Which static analysis tools and Pen Testing services find the CWE's I care about?
Most Important Weaknesses (CWE's)
No single tool can find all of your most important weaknesses –
You Need Multiple Tools!

Figure 14. Covering the Most Important CWEs

Automated SwA tools must be executed on the entire code base, and other assessment approaches may need access to other aspects of the system. Frequently, a contract may cover only an incremental addition to an existing code base or a subsystem. In these cases the SwA assessment approaches used should still be applied to the entire code base and the full system even though the development contract for the new additional section of software code/functionality would only require the repair of defects in the newly written code. The point of assessing all of the system with the software assurance assessment is for the Government to gain insight into the SwA defects in the entire system. This provides the program manager with the insight to existing risk within the software release. In a similar vein, the same issues and responsibilities exist for Foreign Military Sales (FMS) code releases as for U.S. Government use software releases.

The DISA Security Technical Implementation Guides (STIG) Sections 5 and 5.4 [36] advise that automatic static analysis should be applied to all of the source code to identify hidden vulnerabilities and weaknesses in the code base.

4 Deliverables (from EM RFP)

Table 1 is an example list of CDRLs and associated Data Item Descriptions (DID) a program office might include. The selection of the CDRLs can be tailored by the chief engineer based on individual program needs and requirements.

Table 1. CDRL List

CDRL #	CDRL Title	Data Item Description (see ASSIST)
SWA001	Technical Report/Study Services (Software Assurance Evaluation Report)	DI-MISC-80508B
SWA002	Technical Report/Study Services (Software Assurance Case)	DI-MISC-80508B
SWA003	Technical Report/Study Services (Vulnerability Assessment Report)	DI-MISC-80508B
SWA004	Security Test Plan	DI-MISC-8050B
SWA005	Technical Report/Study Services (Security Test Report)	DI-MISC-80508B

4.1 Attachment 1: Software Assurance Evaluation Report Template (from EM RFP)

4.1.1 Assessment Report

- Executive Summary
- Objectives and Technical Scope
- Assessment Approach
- Report of Findings
- Vulnerability Descriptions
- Recommendations for Mitigation

OR

4.1.2 Executive Summary

1 Overview
- 1.1 Objectives
- 1.2 Technical Scope
- 1.3 Participants
- 1.4 Assessment Approach

2 System Overview
3 Security Analysis
  3.1 Critical Findings
  3.2 Major Findings
  3.3 Minor Findings
4 Recommendations
  4.1 Application Risk Assessment Recommendations
  4.2 Detailed Risk Assessment (DRA) Recommendations
  4.3 Software Vulnerability Assessment (SVA) Recommendations
  4.4 Database Security and Compliance Recommendations
Appendix A: Extract of Software Application Tool Scan Results
Appendix B: Extract of Database Tool Scan Results
Appendix C: Security Findings Issue Descriptions
Appendix D: Glossary of Terms

4.2 Software Assurance in CDRL

The CDRL review should occur before software acceptance, and at each delivered software set for evaluation:

- Alpha release
- Beta release
- Release candidate(s)
- Release builds delivered to the Government
- Modify list as needed

The CDRL documents which CWEs, CVEs, CAPECs and software assurance critical violations are absent *and* present in the delivery-candidate source code. This requires the Government to have assessed and defined which CWEs are most important to the project in rank order prior to the CDRL definition. Once this ranked-list of CWEs is created, it can be modified throughout the software development life cycle as needed by the specific program. An example of why a program would change the ordered list of CWE might be the discovery of a new critical CWE that affects the program. This new CWE did not exist during the creation of the first Top-N CWE list for the program but is a critical issue that needs to be resolved. This ranked CWE Top-N list must be placed on contract (with appropriately defined Contract Language). The ranked CWE list is sublisted to include a schedule of:

1. CWEs must not be present in the released software.
2. CWEs should be removed from the released software.
3. CWEs can be removed from the released software.
4. All other CWEs are considered acceptable risks by the Government team.

The above categories can be tied to award fee, or appropriate incentive fee, for the contractor to use in guiding them to produce the highest quality software within the budget and time constraints of the project.

A milestone event should be added to the development milestone list, which occurs after Critical Design Review (CDR) and before Test Readiness Review (TRR) (and software acceptance). This milestone event would contain the following entrance and exit criteria:

- **Entrance Criteria:** The software developer must submit a report/Contract Data Requirements List (CDRL) item to the Government entitled "Software Assurance," which defines how the software was developed and verified for software assurance and provides a detailed report of the Common Weakness Enumerations (CWE), and critical vulnerabilities, present in the delivered software with an adjudication of each CWE, and software assurance critical vulnerability, in the software.
- **Exit Criteria:** The Government validates that all of the CWEs, and software assurance critical violations, as defined by the Government on contract, are absent from the delivered source code. If the software delivered as described in the report satisfies the Government requirements (on contract) then the exit criteria has been met. If the exit criteria have not been met then this milestone remains open and the contractor must repair the software per the Government's direction to remove the noted CWEs and software assurance critical violations and both update the CDRL and resubmit the software and report for evaluation.

The Government is responsible for defining which CWEs, in priority order, are to be absent from the delivered software as well as what constitutes a software assurance critical vulnerability.

5 Incentive and Award Fees

There are several methods to create incentives for software development contractors to produce higher quality code through software assurance. Each method depends on the contract type being used. For example, one method for an award fee contract is to tie the award fee to the proven absence of your specific Top-N CWEs from your Top- 25 CWE list.

> X% of the Award Fee for the Contractor is tied to the proven eradication of all items on the Top-25 list.

• Top-1 through Top-5	==>	0%	of X%
• Top-1 through Top-10	==>	20%	of X%
• Top-1 through Top-15	==>	40%	of X%
• Top-1 through Top-20	==>	60%	of X%
• Top-1 through Top-25	==>	100%	of X%

Acronyms

AFLCMC	Air Force Life Cycle Management Center
BTLJ	Berkeley Technology Law Journal
CAPEC	Common Attack Pattern Enumeration and Classification
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CONOPS	Concept of Operations
CoP	Community of Practice
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DAG	Defense Acquisition Guidebook
DFARS	Defense Federal Acquisition Regulation Supplement
DID	Data Item Description
DISA	Defense Information Systems Agency
DoD	Department of Defense
DPAP	Defense Procurement and Acquisition Policy
DRA	Detailed Risk Assessment
EM	Engineering Model
FAR	Federal Acquisition Regulation
FMS	Foreign Military Sales
GFI	Government-Furnished Information
iEHR	Integrated Electronic Health Record
IT	Information Technology
JFAC	Joint Federated Assurance Center
ODASD(SE)	Office of the Deputy Assistant Secretary of Defense for Systems Engineering
OWASP	Open Web Application Security Project
PPP	Program Protection Plan
RFP	Request for Proposals
SCRM	Supply Chain Risk Management

SDLC	System Development Life Cycle
SEP	Systems Engineering Plan
SOO	Statement of Objectives
STIG	Security Technical Implementation Guide
SVA	Software Vulnerability Analysis
SwA	Software Assurance
TRR	Test Readiness Review
UCC	Uniform Commercial Code
USD(AT&L)	Under Secretary of Defense for Acquisition, Technology, and Logistics
WBS	Work Breakdown Structure

References

[1] National Defense Authorization Act for Fiscal Year 2013 112th Congress. Public Law 112–239, Section 933, January 2013.
https://www.gpo.gov/fdsys/pkg/PLAW-112publ239/pdf/PLAW-112publ239.pdf

[2] USD(AT&L) Memorandum, Document Streamlining–Program Protection Plan (PPP), July 18, 2011.
https://www.acq.osd.mil/se/docs/PDUSD-ATLMemo-Expected-Bus-Practice-PPP-18Jul11.pdf

[3] Program Protection Plan Outline and Guidance version 1.0, July 2011.
https://www.acq.osd.mil/se/docs/PPP-Outline-and-Guidance-v1-July2011.pdf

[4] Defense Acquisition Guidebook Chapter 9, Program Protection.
https://www.dau.mil/tools/dag

[5] Common Attack Pattern Enumeration and Classification (CAPEC™). A publicly available, community-developed list of common attack patterns along with a comprehensive schema and classification taxonomy - to analyze environments, code, and interfaces for common destructive attack patterns, https://capec.mitre.org/.
CAPEC is a catalog of attack patterns along with a comprehensive schema and classification taxonomy focused on enhancing security throughout the software development life cycle, and to support the needs of developers, testers and educators. By providing a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community, CAPEC provides for a more complete and thorough review of the strength of our systems from the point-of-view of attackers.

[6] ITU-T Telecommunication Standardization Sector of ITU, X.1544 Series X: Data Networks, Open System Communications and Security, Cybersecurity information exchange–Event/incident/heuristics exchange, Common attack pattern enumeration and classification, April 2014. http://www.itu.int/rec/T-REC-X.1544-201304-I

[7] Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today, February 2011
http://www.safecode.org/publication/SAFECode_Dev_Practices0211.pdf

[8] Practical Security Stories and Security Tasks for Agile Development Environments, July 2012.
http://safecode.org/publication/SAFECode_Agile_Dev_Security0712.pdf

[9] ISO/IEC TR 20004: 2015, Information technology — Security techniques — Refining Software vulnerability analysis under ISO/IEC 15408 and ISO/IEC 18045, 2015
https://www.iso.org/standard/68837.html

[10] NIST SP800-51 revision 1, Guide to Using Vulnerability Naming Schemes, February 2011.
http://csrc.nist.gov/publications/nistpubs/800-51-rev1/SP800-51rev1.pdf

[11] NIST SP800-53 revision 4, Security and Privacy Controls for Federal Information Systems and Organizations, April 2013.
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[12] Office of the Deputy Assistant Secretary of Defense (ODASD) Systems Engineering (SE) Trusted Systems and Networks (TSN) Analysis, June 2014.
https://www.acq.osd.mil/se/docs/Trusted-Systems-and-Networks-TSN-Analysis.pdf

[13] Office of the Deputy Assistant Secretary of Defense (ODASD) Systems Engineering (SE) Software Assurance Countermeasures in Program Protection Planning, March 2014.
https://www.acq.osd.mil/se/docs/SwA-CM-in-PPP.pdf

[14] Defense Acquisition Guidebook Chapter 9, "Program Protection," August 21, 2017. https://www.dau.mil/tools/dag

[15] Institute for Defense Analyses (IDA) State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation 2016. https://www.acq.osd.mil/se/docs/P-8005-SOAR-2016.pdf

[16] Common Weakness Enumeration (CWE™) - A Community-Developed Dictionary of Software Weakness Types - to examine software architectures, designs, and source code for weaknesses. http://cwe.mitre.org/
Targeted to developers and security practitioners, CWE) is a formal or dictionary of common software weaknesses created to serve as a common language for describing software security weaknesses in architecture, design, or code; serve as a standard measuring stick for software security tools targeting these weaknesses, and to provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

[17] ITU-T Telecommunication Standardization Sector of ITU, X.1524 Series X: Data Networks, Open System Communications and Security, Cybersecurity information exchange – Event/incident/heuristics exchange, Common weakness enumeration, March 2012. http://www.itu.int/rec/T-REC-X.1524-201203-I/

[18] CWE/SANS Top 25 Most Dangerous Software Errors http://cwe.mitre.org/top25/
The Top 25 is a consensus list of the most significant software errors that can lead to serious software vulnerabilities. The errors are dangerous because they frequently will allow attackers to completely take over the software, steal data, or prevent the software from working at all. The Top 25 is the result of collaboration between the SANS Institute, MITRE, and many top software security experts in the US and Europe and leverages experiences in the development of the SANS Top 20 attack vectors and MITRE's CWE.

[19] Common Vulnerabilities and Exposures (CVE®) - The Standard for Information Security Vulnerability Names. http://cve.mitre.org/
International in scope and free for public use, CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVE's common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

[20] ITU-T Telecommunication Standardization Sector of ITU, X.1520 Series X: Data Networks, Open System Communications and Security, Cybersecurity information exchange – Event/incident/heuristics exchange, Common vulnerabilities and exposures, January 2014 http://www.itu.int/rec/T-REC-X.1520-201401-I

[21] Lawrence B. Levy, Suzanne Y. Bell, Software Product Liability: Understanding and Minimizing the Risks, *Berkeley Technology Law Journal* (BTLJ) Volume 5, Issue 1, 1990. http://btlj.org/1990/05/26/volume-5-issue-1-spring-1990/

[22] Michael D. Scott, Tort Liability for Vendors of Insecure Software: Has the Time Finally Come? 67 Md. L. Rev. 425, 2008 http://digitalcommons.law.umaryland.edu/mlr/vol67/iss2/5

[23] AFLCMC/EZC – Engineering Model RFP Language, Hanscom Air Force Base, MA, November 2012.

[24] Director, Defense Procurement and Acquisition Policy, OUSD(AT&L), *Department of Defense COR Handbook*. Director, Defense Procurement and Acquisition Policy, OUSD(AT&L), March 22, 2012. https://www.acq.osd.mil/dpap/cpic/cp/docs/USA001390-12_DoD_COR_Handbook_Signed.pdf

[25] Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)), "Suggested Language to Incorporate System Security Engineering for Trusted Systems and Networks into Department of Defense Requests for Proposals."
https://www.acq.osd.mil/se/docs/SSE-Language-for-TSN-in-DoD-RFPs.pdf

[26] Software Assurance in Acquisition and Contract Language Acquisition and Outsourcing, Volume I, Version 1.1, July 31, 2009.
http://osgug.ucaiug.org/utilisec/Shared%20Documents/Security%20Procurement%20Guidelines%20and%20Language/DHS%20-%20Building%20Security%20In%20-%20Software%20Assurance%20in%20Acquisition%20and%20Contract%20Language.pdf

[27] Guide for Integrating Systems Engineering into DoD Acquisition Contracts, version 1.0, December 11, 2006.
https://www.acq.osd.mil/se/docs/Integrating-SE-Acquisition-Contracts_guide_121106.pdf

[28] DoD Open Systems Architecture Contract Guidebook, version 1.1, June 2013
https://acc.dau.mil/osaguidebook

[29] Software Assurance in Acquisition: Mitigating Risks to the Enterprise, National Defense University Press. Polydys, M. & Wisseman, S., 2009
http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA495389

[30] Recommended Software Assurance Acquisition Language for Space and Missile Systems Center. Headquarters Space and Missile Systems Center, SMC/ENP, September 2015.

[31] USTRANSCOM–Information Assurance/Cyberspace Operations Defense Functional Requirements, Cyber Security Language for Information Technology (IT) Requirements.
http://www.transcom.mil/about/org/tccs/Cyber_Defense_IT_Contract_Language.pdf

[32] DoD/VA integrated Electronic Health Record (iEHR) Technical Specifications Summary, sections 6.5.1 through 6.5.7, Version 2.2, June 17, 2013.

[33] Open Web Application Security Project, OWASP Secure Software Contract Annex,
https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex

[34] Deputy Assistant Secretary of Defense for Systems Engineering and Department of Defense Chief Information Officer, Software Assurance Countermeasures in Program Protection Planning, March 2014. https://www.acq.osd.mil/se/docs/SwA-CM-in-PPP.pdf

[35] Integrated Analysis and Reporting in Multiple Tools: What Cybersecurity and Robustness Testing Tool Manufacturers Should Be Building Towards, 2015.
https://interact.gsa.gov/sites/default/files/Mon AM2-Integrated Analysis and Reporting In Multiple Tools V 8-30-2015_MVE -RAM.pdf

[36] DISA Application Security and Development Security Technical Implementation Guide (STIG), Version 3, Rel.10," January 23, 2015.
http://iase.disa.mil/stigs/app-security/app-security/Pages/index.aspx

Incorporating Software Assurance into Department of Defense Acquisition Contracts

Working paper, November 2017

Deputy Assistant Secretary of Defense
Systems Engineering
3030 Defense Pentagon
3C167
Washington, DC 20301-3030
osd.atl.asd-re.se@mail.mil
www.acq.osd.mil/se