Trusted Systems and Networks (TSN) Analysis

JUNE 2014

Deputy Assistant Secretary of Defense for Systems Engineering
and Department of Defense Chief Information Officer

Washington, D.C.

Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)) and Department of Defense Chief Information Officer (DoD CIO). 2014. *Trusted Systems and Networks (TSN) Analysis*. Washington, D.C.: DASD(SE) and DoD CIO.

Office of Primary Responsibility:

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

Contents

Tables

Figures

1 Introduction

This document is intended as an extension to guidance provided in the Defense Acquisition Guidebook (DAG) Chapter 13, Program Protection. This document provides further details for Trusted Systems and Networks (TSN) analysis processes, methods, and tools. It elaborates on each of the major iterative processes necessary to accomplish the TSN analysis objectives.

The TSN analysis consists of several activities (Figure 1-1): a criticality analysis (CA) to determine the most critical functions of the system, a threat assessment to understand the likely attacks, a vulnerability assessment to recognize vulnerabilities in the design and the commercial off-the-shelf (COTS) products, a risk assessment, and selection of security countermeasures (risk mitigations) based on a cost-benefit trade-off analysis. When the selected security countermeasures are planned for implementation into the system, the system's supply chain, and the system's development environments, the risk is reassessed.

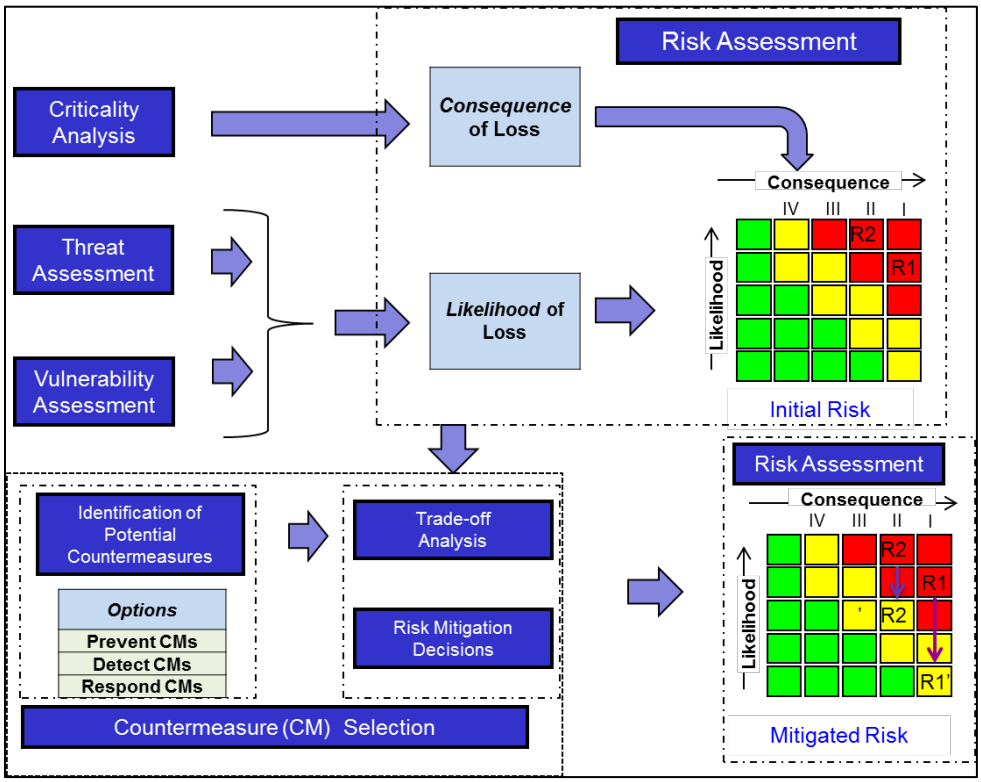

Figure 1-1. Trusted Systems and Network Analysis Methodology

A program should repeat the TSN analysis as the system is refined to respond to a dynamically changing threat environment, the discovery of new vulnerabilities in COTS products, or introduction of new vulnerabilities from design decisions and selection of COTS products. As a minimum the TSN analysis should be updated before each Systems Engineering Technical Review (SETR). The following sections describe each of the TSN analysis steps in detail.

2 Criticality Analysis

The Criticality Analysis allows a program to focus attention and resources on the system capabilities, mission-critical functions, and critical components that matter most. Mission-critical functions are those functions of the system that, if corrupted or disabled, would likely lead to mission failure or degradation. Mission-critical components are primarily the elements of the system (hardware, software, and firmware) that implement mission-critical functions. It can include components that perform defensive functions which protect critical components, and components that have unmediated access to critical components.

The CA is the primary method by which a program identifies mission-critical functions and associated critical components. The CA includes the following iterative steps:

- Identify and group mission threads.
- Decompose the mission threads into their mission-critical functions and assign them criticality levels.
- Map the mission-critical functions to the system architecture and identify the defined system components (hardware, software, and firmware) that implement those functions (i.e., components that are critical to the mission effectiveness of the system or an interfaced network).
- Allocate criticality levels to those components that have been defined.
- Identify suppliers of critical components

The identified functions and components are assigned levels of criticality commensurate with the consequence of their failure on the system's ability to perform its mission, as shown in Table 2-1.

Table 2-1. Protection Failure Criticality Levels

Level I Total Mission Failure	Failure that results in total compromise of mission capability
Level II Significant/Unacceptable Degradation	Failure that results in unacceptable compromise of mission capability or significant mission degradation
Level III Partial/Acceptable	Failure that results in partial compromise of mission capability or partial mission degradation
Level IV Negligible	Failure that results in little or no compromise of mission capability

Source: DAG Chapter 13, Table 13.2.3.1.T1.

A CA typically requires multiple iterations. The first iteration identifies the primary critical functions. The second iteration, usually completed in conjunction with the vulnerability assessment, identifies those functions that have unmediated access to the critical functions. These functions have the same level of criticality as the functions they access.

The third iteration identifies those functions upon which the critical functions depend (e.g. if a critical function depends on a particular software library, that library is also critical). These functions are also critical at the same level as the dependent function.

When identifying critical functions, associated components and their criticality levels, programs should consider the following:

- Information and Communications Technology (ICT) components are especially susceptible to malicious alteration throughout the program life cycle.
- Functional breakdown is an effective method to identify functions, associated critical components, and supporting defensive functions.
- Dependency analysis should be used to identify those functions on which critical functions depend, which themselves become critical functions (e.g. defensive functions and initialization functions).
- The program should identify all access points to protect unmediated access to critical function components (e.g. implement least privilege restrictions).

Once the program has identified critical functions through the CA, the program systems engineers and SSEs can use the results along with the vulnerability assessment and threat assessment to determine the risk.

A DoD program needs to perform CA throughout the acquisition life cycle. At a minimum, DoD programs need to perform / update a CA, along with the threat assessment, vulnerability assessment, risk assessment, cost-benefit trade-off analysis and countermeasure selection, before each SETR.

2.1 Performing the Criticality Analysis

While the Government should perform an initial CA during the Materiel Solution Analysis (MSA) phase, it may only be possible for the program to execute the CA process given below (Table 2-2) at a high level. To be effective, CAs must be executed iteratively across the acquisition life cycle, building on the growing system maturity, knowledge gained from prior CAs, updated risk assessment information, and updated threat and vulnerability data.

For example, the first pass through the CA process, together with assessments of vulnerabilities, threats, risks, and countermeasures, might take just a few days and provide a preliminary result. This CA might involve Subject Matter Expert (SME) viewpoints provided during several work sessions (to address system and architecture), as opposed to detailed information collected from numerous program documents. For an early iteration, precision is not possible, as it takes several iterations to complete the initial CA.

Table 2-2 contains the detailed steps to perform a CA.

Table 2-2. Criticality Analysis Steps

Identify Missions and Mission-Essential Functions	**Sources of Information**
1. Identify mission threads and principal system functions. • Derived first during pre-Milestone A and revised as needed for successive development milestones.	Joint Capabilities Integration and Development System (JCIDS) Documents: • Initial Capabilities Documents (ICD) • Capability Development Documents (CDD) • Capability Production Documents (CPD) Concept of Operations
2. If possible or necessary, group the mission capabilities by relative importance. Training or reporting functions may not be as important as core mission capabilities.	Operational Representative Subject Matter Expertise (Integration Experts, Chief Engineers)
3. Identify the system's mission-critical functions based on mission threads and the likelihood of mission failure if the function is corrupted or disabled. (Mission-critical functions may include navigating, targeting, fire control, etc.).	Activity Diagrams Use Cases Functional Decomposition Potential Department of Defense Architecture Framework (DODAF) Sources • OV-5 (Operational Activity Model) • SV-4 (System Functionality Description) Subject Matter Expertise
Identify Critical Subsystems, Configuration Items, and Components	
4. Map the mission threads and functions to the system architecture and identify critical subsystems, Configuration Items (CI), and sub-CIs (components). Note: Focus on CIs and components containing Information and Communications Technologies (ICT). Logic-bearing components have been singled out as often implementing critical functions and as susceptible to life cycle corruption.	System/Segment Design Document Architecture Description Document Requirements Traceability/Verify Matrix Potential DODAF Sources • SV-5a (Operational Activity to System Function Traceability Matrix)
5. Assign levels of criticality (I, II, III, IV) to the identified CIs or components. Factors or criteria may include: • Frequency of component use across mission threads • Presence of redundancy; triple-redundant designs can indicate critical functions. Subject matter expertise	Subject Matter Expertise • Systems Engineer • Operators Representative • Program Office
6. Identify any CIs or components that do not directly implement critical functions but either have unmediated communications access (i.e., an open access channel) to one or more critical functions or protect a critical function. • Which components give or receive information to/from the critical components? Note: A non-critical component may communicate with a critical function in a way that exposes the critical function to attack. In some cases, the architecture may need to include defensive functions or other countermeasures to protect the critical functions.	Architecture Diagrams Subject Matter Expertise Data Flow Diagram

Initial Start Conditions	
7. Identify critical conditions/information required to initialize the system to complete mission-essential functions. • What information is needed to successfully execute capabilities? How is this information obtained, provided, or accessed by the system? • How quickly must information be received to be useful? Does the sequence in which the system initializes itself (power, software load, etc.) have an impact on performance?	Data Flow Diagram Information Support Plan
8. Based on the answers to the questions above, identify these functions or components to be included in program protection risk management.	
Operating Environment	
9. Identify the system functions or components required to support operations in the intended environment. These may include propulsion (the system has to roll, float, fly, etc.); thermal regulation (keep warm in space, keep cool in other places, etc.); or other environmentally relevant subsystems that must be operational before the system can perform its missions.	Architecture Diagrams
10. Identify the ICT implementing those system functions and any associated vulnerabilities with the design and implementation of that ICT.	
Critical Suppliers	
11. Identify suppliers of critical configuration items or ICT components.	Manufacturing Lead
Note: Repeat this process as the system architecture is refined or modified, such as at Systems Engineering Technical Reviews and major acquisition milestone decision points. • Design changes may result in adding or removing specific CIs and sub-CIs from the list of critical functions and components.	

CDD: Capability Development Document
CI: Configuration Item
DODAF: DoD Architecture Framework
ICD: Initial Capabilities Document
ICT: Information and Communications Technology
JCIDS: Joint Capabilities Integration and Development System
Sub-CI: Sub-Configuration Item

The program should consider the following when carrying out the CA:

- Document the results of each step.
  - Include rationale.
- Use questions to support the analysis; for example:
  - What information is needed to successfully execute capabilities?
  - How is this information obtained, provided, or accessed by the system?
  - How quickly must information be received to be useful?
  - Does the sequence in which the system initializes itself (power, software load, etc.) have an impact on performance; for example, areas in which a specific sequence may have an impact:
    - Propulsion (the system has to roll, float, fly, etc.)
    - Thermal regulation (keep warm in space, keep cool in other places, etc.)
    - Other environmentally relevant subsystems that must be operational before the system can perform its missions
- Use available artifacts to inform the CA; for example:
  - Systems engineering artifacts such as architectures/designs and requirements traceability matrices
  - Available threat and vulnerability information
  - Residual vulnerability risk assessments to inform follow-up CAs
- In isolating critical functions/components, identify critical conditions/information required to initialize the system to complete mission-critical functions.
- Identify the subsystems or components required to support operations in the intended environment.

2.2 Analysis Results

The expected output of an effective CA process is:

- A complete list of mission-critical functions and components
- Criticality level assignments for all items in the list
- Rationale for inclusion or exclusion from the list
- Supplier information for each critical component
- Identification of critical elements for inclusion in a Defense Intelligence Agency (DIA) Threat Assessment Center (TAC) Request.

The identification of critical functions and components and the assessment of system impact if compromised is documented in the Program Protection Plan (PPP) as discussed in Appendix C (Table C-1) of the PPP Outline.

The prioritization of Level I and Level II components for expending resources and attention will be documented in the PPP as discussed in Appendix C (Table C-2) of the PPP Outline.

The Level I and selected Level II components from the CA are used as inputs to the threat assessment, vulnerability assessment, risk assessment, and countermeasure selection. The following sections describe these activities.

3 Threat Assessment

Programs utilize all-source intelligence to understand the threats to the system and the threats posed by specific suppliers. Multiple sources of intelligence can be used to feed into this analysis. In the absence of threat information, a program should assume a medium or high threat, in order to avoid missing a window for implementing cost-effective countermeasures. If no threat is assumed, and then threat information becomes available, indicating a high threat, the cost to mitigate the risk posed by the threat may be prohibitively costly.

One specific source for supplier threat information is the Defense Intelligence Agency (DIA) Threat Analysis Center (TAC). DoD has designated the DIA to be the DoD enterprise responsible entity for threat assessments needed by the DoD acquisition community to assess supplier risks. DIA established the TAC for this purpose.

3.1 DIA Supply Chain Risk Management Threat Assessment Center

DoD has designated the Defense Intelligence Agency (DIA) as the DoD enterprise responsible for threat assessments needed by the DoD acquisition community to assess supplier risks. The TAC provides the enterprise management and interface to resources within the National Counterintelligence Executive, and coordinates with the Defense Intelligence and Defense Counterintelligence Components to provide standard all-source intelligence assessments to support acquisition risk management efforts. The TAC's enterprise role was intended to allow the Department to achieve comprehensive and consistent supplier threat assessments across the Military Departments and Defense Agencies and to ensure the efficient use of the results by the acquisition community.

DIA threat assessments provide specific and timely characterization of threats associated with the identified suppliers. Program managers and engineering teams consider TAC reports when selecting supplier and/or architecture alternatives and developing appropriate mitigations for supply chain risks. For the policy and procedures regarding the request, receipt, and handling of TAC reports, refer to DoD Instruction (DoDI) O-5240.24, "Counterintelligence (CI) Activities Supporting Research, Development, and Acquisition (RDA)."

Requests for TAC assessments are developed based on the CA. To determine which requests are needed, the program may refer to an annotated work breakdown structure (WBS) or system breakdown structure (SBS) that identifies the suppliers of critical functions and components. The program may submit requests for TAC assessments as soon as sources of critical capability are identifiable.

Near the end of the MSA phase, as some threat information is available from the capstone threat assessment (CTA) and technologies and potential suppliers are identified, the program may use TAC assessments to assist in defining lowest risk architectures, based on suppliers for particular architecture alternatives. Early in the system life cycle, the threat requests may be more focused on suppliers in general technology areas to inform architecture choices, whereas later in the system life cycle the requests may be more focused on critical components defined in the CA.

Engineering activities related to SCRM begin as the program considers architecture alternatives and continue throughout the acquisition life cycle. As the systems engineering team develops the initial view of system requirements and system design concepts, the team performs a CA to define critical technology elements. CA produces a list of critical components and suppliers that are used to generate TAC requests and supplier risk mitigation.

The program continues to update and enhance the CA through the Full-Rate Decision and sustainment, incorporating more details as architecture decisions are completed and the system boundaries are fully defined. The engineering team may at any point, beginning prior to Milestone A, identify technology elements and potential manufacturers and request supplier threat assessments. The number of supplier threat assessment requests will grow as the CA becomes more specific and the system architecture and boundaries are fully specified; in other words, the greatest number of TAC requests will typically occur between Milestones B and C (i.e., PDR and CDR).

4 Vulnerability Assessment

This section describes a process for identifying vulnerabilities in systems, supply chains and development and test environments. A vulnerability is any weakness in system design, development, production, or operation that can be exploited to defeat a system's mission objectives or significantly degrade its performance. Vulnerability assessment is one step in the TSN Analysis.

An adversary that is able to gain access to, change, or limit a system's performance is extremely dangerous. Different weaknesses and vulnerabilities that could allow an adversary to impact the mission need to be assessed. Decisions about which vulnerabilities need to be addressed and which countermeasures or mitigation approaches to apply are based on an overall understanding of threats, impact to the mission, and program priorities.

4.1 Approaches to Identifying Vulnerabilities

Throughout a system's design, development testing, production, and maintenance, a program should be aware of vulnerabilities that enable malicious activities that could interfere with the

system's operation. Vulnerabilities identified early in a system's design often can be eliminated with simple design changes or procurement constraints at relatively low cost. Vulnerabilities addressed later may require add-on protection measures or operating constraints that may be less effective and more expensive.

The principal vulnerabilities to watch for in an overall review of systems engineering processes are

- Access paths within the supply chain, development and test environments and processes that would allow adversaries to introduce components (hardware, software, and firmware) that could cause the system to fail at some later time;
- Access paths that would allow threats to trigger a component malfunction or failure at a time of the adversary's choosing; and
- Access paths within the architecture and design that allow threats to circumvent the integrity, confidentiality, and availability of the mission system through weaknesses in the component design, architecture, or code.

Supply chain here means any point in a system's design, engineering, and manufacturing development, production, configuration in the field, updates, and maintenance. Access opportunities may be for extended or brief periods. The need to protect the supply chain and development environments extends the vulnerability assessment beyond the system to the program processes and tools used to obtain and maintain the hardware, software and firmware components of the system.

The following six techniques and tools have proven effective in identifying vulnerabilities:

- Milestone A vulnerability assessment questionnaire
- Vulnerability databases
- Static analyzer tools and other detection techniques
- Component diversity analysis
- Fault Tree Analysis (FTA)
- Red team penetration testing

A program may use several of these techniques to have a full life cycle approach to the vulnerability assessment.

4.1.1 Milestone A Vulnerability Assessment Questionnaire

The Milestone A Vulnerability Assessment Questionnaire (Appendix A) is a set of yes or no questions that a program answers to identify vulnerabilities in the Statement of Work (SOW) and System Requirements Document (SRD) before RFP release. Appendix A includes the procedure for using the questionnaire and applying the results to determine the system security risk likelihood.

4.1.2 Vulnerability Databases

This assessment approach uses three databases of publicly available information that define attack patterns, weaknesses, and vulnerabilities: the Common Attack Pattern Enumeration and Classification (CAPEC) [1], the Common Weakness Enumeration (CWE) [2], and the Common Vulnerabilities and Exposures CVE) [3].

The CAPEC is a resource to identify the attack patterns that an adversary might attempt to use against a system. By reviewing the types of vulnerabilities that different attack patterns are effective in attacking, a program can identify vulnerabilities in its own system. The CAPEC lists potential attacks on the system as well as on the supply chain and the development environments, including the personnel involved in those activities. A program should select all the attacks that are effective against the weaknesses the system could have and that could cause an undesirable mission impact.

Using the set of attack vectors, the program then finds the set of weaknesses associated with the attacks in the CWE and the CVE databases and uses these to evaluate the development software, legacy software, open source, and COTS software. The program uses the weaknesses and vulnerabilities, aligned to the software life cycle and stage of development, to complete the vulnerability assessment. The program then uses the vulnerability assessment as the basis for inspecting the design and architecture, the developed software, COTS software, and the deployed environment in which it will operate.

The CVE database contains publicly known vulnerabilities that need to be patched or otherwise remediated. The program should use the CVE to review the commercial and open source components in a system, in the development environment, and in the test environment to address vulnerabilities in all aspects. The CWE lists the types of vulnerabilities (weaknesses) that can occur in software processes, practices, design, the architecture, the code, or the deployed instance of the software. Appendix B further describes and illustrates the assessment technique.

4.1.3 Static Analyzer Tools and Other Detection Methods

For software systems, a program can use static analysis, dynamic analysis, and other testing, tools, and techniques to identify vulnerabilities in software during development, in legacy software, and in open source. Many static and dynamic analysis tools and security analysis service offerings relate the vulnerabilities to specific CWE weaknesses and specific CVE vulnerability entries. Static and dynamic analyzers from different vendors use different testing techniques and internal criteria and often find different weaknesses and vulnerabilities.

Before making use of a static or dynamic analyzer or the services of a security assessment team, the program needs to define the categories of defects to be addressed and review which can be found by which detection method and capabilities offered. For those capabilities that relate the defects to specific CWE and CVE entries, the results can be combined with the Vulnerability Database Assessment described in 4.1.2.

4.1.4 Component Diversity Analysis

A program can use component diversity analysis to assess the potential impact of malicious insertion in a component that is used multiple times in one or more critical functions or subfunctions.

As the system design is developed and refined, various factors can affect the program's selection of components. One of these factors is commonality. If a similar type of component is needed in multiple places within or across subsystems, selecting common components is potentially advantageous in terms of maintainability, reliability, and life cycle cost. For example, common components can lower cost by allowing for economies of scale or lower spare part inventories.

However, common components can increase the system security risk. If a common component is used multiple times within or across critical functions or subfunctions, the vulnerabilities of that particular component also are common across the functions. It makes the component a higher value target for malicious insertion of logic because the impact of exploiting a particular vulnerability is increased.

One potential way to mitigate this risk is through component diversity. Adding design and component diversity into the system lowers the impact of exploiting a particular vulnerability. For example, a microprocessor needed in three separate subsystems to implement a critical function in each subsystem will affect three critical functions if a common component is selected. If two different microprocessors are chosen, although each will potentially have vulnerabilities, an exploitation of a single vulnerability will not affect all three critical functions. Similarly in cases where reliability dictates the need for redundancy use of diverse redundancy will add security to the system.

There is also the potential to apply diversity to the supply chain. If choosing the same component is the only practical measure, consider using multiple sources to supply the component. For the microprocessor example, having multiple sources lowers the likelihood that both components will have been subverted in the supply chain.

When assessing component diversity, it is important to balance the security benefits of diverse components with the potential cost savings of common components. This analysis may be performed at the subsystem, system, or system-of-systems levels to ensure use of component diversity across our systems.

Component diversity analysis is one way of assessing the potential impact a vulnerability may have on a system on a larger scale. This assessment can be completed at various points in the life cycle. Earlier in the life cycle, these may be notional/preliminary components, while the components may be more finalized when developing the Allocated Baseline. Earlier life cycle iterations allow for designing in component diversity.

4.1.5 Fault Tree Analysis

Fault Tree Analysis (FTA) is a technique commonly used in system safety and reliability engineering to discover how systems might fail and to find ways to reduce the number and severity of safety incidents and system downtime. FTA is also applicable to system security

engineering (SSE), with some adjustments to account for malicious actors introducing intentional system faults, as opposed to random sources of failures.

FTA is a top-down approach that uses Boolean logic to identify potential sources of system failures. FTA assumes a hypothetical system or mission failure has occurred, and traces that outcome back through the system to determine contributing component malfunctions or failures. At the top level, a fault tree for system security might look something like the diagram shown in Figure 4-1.

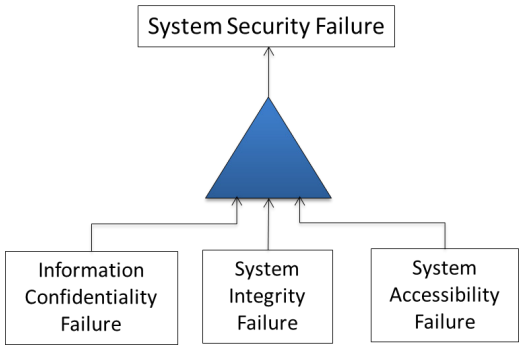

Figure 4-1. Example Top-Level Fault Tree Diagram

The triangle in this diagram indicates a logical "OR" function that combines three common types of security failures. Each of the component failure categories would be broken down further, for example to show that a confidentiality breach would require both access to information that should be protected "AND" some means for transferring that information into the hands of an untrusted party. The diagram would then be expanded further to identify access and data transfer paths within the system, looking for possible combinations that would allow a breach to occur.

In addition to tracing hypothetical security breaches throughout the system architecture, a thorough FTA should consider possible access paths and opportunities an adversary might use to discover vulnerabilities or introduce new ones into the system. Access to design or software development information could provide valuable information for planning cyber attacks. If an attacker can intercept and substitute or modify component products before they reach the system integrator, then the attacker can introduce new vulnerabilities into the supply chain. These additional sources of security problems are not usually considered in safety or reliability analyses, but they are important considerations for protecting the system against malicious threats.

To scope the engineering effort, time, and cost of an FTA, the program must consider the breadth and depth of analysis needed to safeguard mission-critical functions. The program should focus on mission-critical tasks, critical programmable and logic-bearing components, and any uncontrolled access (including network access) that might provide an intrusion path. For example, an FTA approach would include the following activities:

- Establish the set of failure events to be evaluated based upon the list of critical functions.
- For each failure event, decompose the fault tree to identify the logical dependencies among hypothetical component failures.

- Identify any "hot spots" of components that represent significant risks because they play a role in multiple failure events.

4.1.6 Red Team Penetration Testing

"Penetration testing is the simulation of an attack on a system, network, piece of equipment or other facility, with the objective of proving how vulnerable that system or "target" would be to a real attack" (Henry 2012). Red teams typically subject a system, supply chain, and the development environment to a series of attacks, simulating the tactics of an actual threat. The basic approach is to gather data about the system, supply chain and development environment and to define the objectives, type of attacks, and scope of the attacks. The types of attacks are a set of abuse or misuse cases that can be defined in a manner similar to use cases. Probing and failed attacks contribute to extending the knowledge of the security behavior of the system, supply chain, and development environment.

4.2 Identifying Potential Vulnerability Mitigations or Countermeasures

Multiple countermeasures are available to mitigate a range of possible vulnerability risks. Design changes may (1) eliminate exploitation, (2) reduce the consequences of exploitation, or (3) block the access necessary for introduction or exploitation. Add-on protection mechanisms may block the access required to trigger exploitation. An effective update process, particularly for software, can correct or counteract vulnerabilities discovered after fielding.

It is unlikely a program can completely prevent exploitation of vulnerabilities. As a result, a balanced approach to countermeasures should include prevention, detection (monitoring), and response. Anonymous purchases (blind buys) COTS may prevent an untrustworthy supplier from knowing where the component is being used. More extensive testing may be required for critical components from unverified or less dependable sources to detect malicious insertion of logic. A variety of different countermeasures should be identified to inform and provide options for the program manager's risk-mitigation decisions.

4.3 Interactions with Other Program Protection Processes

Investigation of vulnerabilities may indicate the need to raise or at least reconsider the criticality levels of functions and components identified in earlier criticality analyses. Investigation of vulnerabilities may also reveal additional threats, or opportunities for threats, that were not considered risks in earlier vulnerability assessments. Vulnerabilities inform the risk assessment and the countermeasure cost-risk-benefit trade-off. Threat assessments can inform vulnerability assessments by identifying attack paths and areas of particular interest.

5 Trusted Systems and Networks Risk Assessment

For each Level I and Level II critical function or component, the program performs a risk assessment. Figure 1-1 (page 1) shows how risk assessment is performed in the context of the TSN analysis.

5.1 Determining Consequence

Consequence is determined based on the results of the CA. The program uses the system impact level from the CA to determine the risk consequence.

5.2 Determining Likelihood

The risk likelihood is based on the vulnerability assessment and the threat assessment. Each Service and program may have specific guidance on how to use the assessments to develop the risk likelihood. One approach is to average the two likelihoods. Another approach is to use the higher of the two likelihoods for the risk cube.

Each of the recommended techniques for the vulnerability assessment includes a way to characterize the vulnerability likelihood. All of the techniques address known vulnerabilities. The more vulnerabilities that are identified, the higher the risk and the more likely there will be a successful attack.

Techniques to estimate the likelihood of unknown vulnerabilities are just beginning to emerge.

5.3 Determining Risk

Once the consequence and likelihood are determined, the risk can be represented on a risk cube. The risk is then incorporated into the program technical risks. The risk entry may look similar to the example shown in Table 5-1. The program must establish a risk cube and mitigation plans for all top program protection risks (very high and high).

Table 5-1. Risk Likelihood After Mitigations

Software Assurance Technical Risks		**Possible Mitigation Activities**
R1. Field-programmable gate array (FPGA) 123 has high exposure to software vulnerabilities with potential foreign influence		Establishing a wrapper to implement secure design standards and fault logging, static analysis, increased test coverage, and penetration testing
Technical Issues		
1. May impact performance, cost, and schedule		
Opportunities		
O1. Significant investment, increased security for program, and overall for missile domain of programs		Significant investment but secure operation and system dependability, benefit to program and command

6 Countermeasure Selection

This section describes the guidance and expectations for TSN countermeasures. Countermeasures are cost-effective activities and attributes to manage risks to critical functions and components. They vary from process activities (e.g., using a blind buying strategy to obscure the end use of a critical component) to design attributes (e.g., interface input and output checking to ensure the component is operating within specification) and are selected to mitigate a particular risk. For each countermeasure being implemented, the program identifies the person responsible for its execution and a time- or event-phased plan for implementation.

Many countermeasures may have to be partially or completely implemented by prime and subcontractors on the program. See "Suggested Language to Incorporate System Security Engineering for Trusted Systems and Networks into Department of Defense Requests for Proposals" (DASD(SE) 2014), http://www.acq.osd.mil/se/initiatives/init_pp-sse.html for guidance on contracting for the implementation of program protection.

A balanced approach to countermeasures should include prevention, detection (monitoring), and response countermeasures.

- Prevent – Countermeasures that reduce the exploitation of development, design, and supply chain vulnerabilities
- Detect – Countermeasures that monitor, alert, and capture data about the attack
- Respond – Countermeasures that analyze attacks and alter system or processes to mitigate the attack

The early phase PPPs should contain all three types of countermeasures as well as plans for more detailed program protection analysis and updates to inform SSE early in the design.

6.1 Risk Cost-Benefit Trade

Programs complete system security risk cost-benefit trade-off analysis to develop a set of process requirements, system security requirements, constraints, and design attributes to be included in the system baseline. Information for the trade-off analysis includes the comprehensive set of countermeasures (comprehensive means countermeasures that detect and respond to attacks as well as countermeasures that prevent attacks) and the results of the vulnerability assessment, threat assessment, and cybersecurity (information assurance) assessment. The trade-off analysis results in a set of countermeasure requirements to be incorporated into the system requirements baseline and the SOW.

The systems engineer needs to recognize that vulnerabilities will continue to be identified during the system development and operation, and thus the system security requirements will need to be reassessed and updated as system requirements and design decisions are made. To develop this set of SSE requirements, the program systems engineer and the system security engineer perform two levels of trade-off analysis: a security domain-level analysis and a system-level analysis.

In the security domain-level analysis, the system security engineer trades potential countermeasures to identify a cost-effective set of system security requirements. In the system-level analysis, the systems engineer considers the broader system functional and non-function performance requirements and design characteristics to ensure a balanced trade-off of system security requirements versus performance and cost requirements.

This two-tiered analysis leads to a dynamic environment in which systems engineering trade-offs outside of the security domain may trigger a need to update the SSE analysis and trade-offs. To conduct trade-off analysis for the supply chain, development processes, and the development tools, systems engineers must interact with procurement and acquisition personnel as the program establishes the system specification and design.

Risk, cost, and benefit factors influence these two levels of trade-offs. The systems engineer may explore alternative designs to evaluate the new or revised requirements. The output of this step is a set of affordable countermeasure requirements to be incorporated into the system requirements baseline and acquisition-process requirements to be incorporated into the SOW.

7 Relationship Between TSN and Other System Security Processes

7.1 Relationship with DoD Risk Management Framework for Information Technology

DoD's instantiation of the Risk Management Framework (RMF) is presented in DoDI 8510.01, "Risk Management Framework (RMF) for DoD Information Technology (IT)." This policy replaces the former DIACAP (Defense Information Assurance Certification and Accreditation Process) the DoD used for certification and accreditation. The policy manages the life cycle cybersecurity risk to DoD IT, including Information Systems and Platform IT (PIT) systems. Although the main tenets of both the NIST RMF and the DoD RMF are the same, there are some differences in the policies and how they are executed. More information can be found in the DoDI 8510.01 and the RMF Knowledge Service (https://rmfks.osd.mil).

There are several places in which the DoD RMF activities interact with the steps of the DoD Risk Management Process. For example, the DoD RMF prescribes selecting applicable security controls for the system. Once the program has established a control baseline using system categorization and has assigned relevant overlays, the program can tailor that control set based on the results of the threat assessment, vulnerability assessment, and cybersecurity (information assurance) assessment, which feed into a risk analysis and a cost-benefit trade analysis. The program may add security controls to the RMF or may trade controls away in light of other mitigating factors. Also the program may define different levels of strength of implementation in order to mitigate identified risks to acceptable levels.

The program tailors the set of security controls and translates the controls into requirements and design details to ensure the controls mitigate vulnerabilities to confidentiality, integrity, and availability. The program captures the control requirements in the system requirements and functional baselines to ensure the control requirements are implemented and traced throughout the design and development of the system.

Although each step in the Risk Management Process is repeated throughout the system life cycle, the DoD RMF calls out a specific vulnerability assessment in its Step 4: Assess Security Controls. During this step the program prescribes an independent assessment of each security control's compliance, deficiency level, risk level, and remediation activities, and then the associated cybersecurity risk to the system. This analysis is unique because it examines cybersecurity vulnerabilities to the system in the context of the security controls once they have been implemented. From a cybersecurity perspective, this analysis helps a program determine the implementation of a standard set of cybersecurity requirements using a common process.

7.2 Relationship with Critical Program Information Identification and Protection

End-items identified as critical program information (CPI) generally perform a function that gives the United States a capability advantage. Therefore, an end-item identified as CPI is probably performing a critical function and may be identified as a Level I or II critical component.

When developing countermeasure(s) for this situation, countermeasures applicable to CPI and countermeasures applicable to critical components must both be considered. Certain countermeasures may be applicable to reduce the risk from both the TSN risk and the CPI risk and therefore may offer a more affordable solution.

References

Defense Acquisition Guidebook (DAG). Washington, D.C.: Under Secretary of Defense for Acquisition, Technology, and Logistics. https://dag.dau.mil/.

Department of Defense (DoD). 2006. "Risk Management Guide for DoD Acquisition." 6th ed. (Version 1.0). Washington, D.C.: Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics/Systems and Software Engineering (now Deputy Assistant Secretary of Defense for Systems Engineering).

Department of Defense Instruction (DoDI) 5000.02 (Interim). 2013. "Operation of Defense Acquisition System." Washington, D.C.: Under Secretary of Defense for Acquisition, Technology, and Logistics (November 25). http://www.dtic.mil/whs/directives/corres/pdf/500002_interim.pdf

Department of Defense Instruction (DoDI) 5200.44. 2012. "Protection of Mission-Critical Functions to Achieve Trusted Systems and Networks." Washington, D.C.: DoD Chief Information Officer/Under Secretary of Defense for Acquisition, Technology, and Logistics (November 5). http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf

Department of Defense Instruction (DoDI) O-5240.24. 2011. "Counterintelligence (CI) Activities Supporting Research, Development, and Acquisition (RDA)." Washington, D.C.: Under Secretary of Defense for Intelligence.

Department of Defense Instruction (DoDI) 8510.01. 2014. "Risk Management Framework for DoD Information Technology (IT)." Washington, D.C.: DoD Chief Information Officer.

Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)). 2014. "Suggested Language to Incorporate System Security Engineering for Trusted Systems and Networks into Department of Defense Requests for Proposals." Washington, D.C.: DASD(SE). http://www.acq.osd.mil/se/initiatives/init_pp-sse.html

Henry, Kevin M. 2012. "Penetration Testing: Protecting Networks and Systems." Ely, Cambridgeshire, United Kingdom: IT Governance Publishing. http://www.itgovernanceusa.com/

National Defense Industrial Association (NDIA) System Assurance Committee. 2008. *Engineering for System Assurance.* Arlington, Va.: NDIA. http://www.acq.osd.mil/se/docs/SA-Guidebook-v1-Oct2008.pdf

National Institute of Standards and Technology (NIST). 2012. System Security Engineering Processes. Draft of forthcoming Special Publication 800-160. Washington, D.C.: NIST, U.S. Department of Commerce.

Appendix A: Milestone A Vulnerability Assessment Questionnaire

Part I – Supply Chain Vulnerabilities

1. ___ Does the Statement of Work (SOW) require the contractor to have a process to establish trusted suppliers?

2. ___ Does the SOW require the contractor to obtain DoD-specific Application-Specific Integrated Circuits (ASICS) from a Defense Microelectronics Activity (DMEA)-approved supplier?

3. ___ Does the SOW require the contractor to employ protections that manage risk in the supply chain for critical components or subcomponent products and services (e.g., integrated circuits, field-programmable gate arrays (FPGA), printed circuit boards) when they are identifiable (to the supplier) as having a DoD end-use?

4. ___ Does the SOW require the contractor to require suppliers to have similar processes for the above questions?

5. ___ Does the SOW require the prime contractor to vet suppliers of critical function components (hardware/software/firmware) based upon the security of their processes?

6. ___ Does the SOW require the contractor to use secure shipping methods for critical components? How are components shipped from one supplier to another?

7. ___ Does the SOW require the contractor to have processes to verify critical function components received from suppliers to ensure that components are free from malicious insertion (e.g., seals, inspection, secure shipping, testing, etc.)?

8. ___ Does the SOW require the contractor to have controls in place to ensure technical manuals are printed by a trusted supplier who limits access to the technical material?

9. ___ Does the SOW require the contractor to have controls to limit access to critical components?

10. ___ Does the SOW require the contractor to identify everyone that has access to critical components?

11. ___ Does the SOW require the contractor to use blind buys to contract for [selected] critical function components?

12. ___ Does the SOW require specific security test requirements to be established for critical components?

13. ___ Does the SOW require the developer to define and use secure design and fabrication or manufacturing standards for critical components?

Part II – Software Development Vulnerabilities

1. ___ Does the SOW require the contractor to establish secure design and coding standards for critical function components developmental software (and that are verified through inspection or code analysis)?
   - The contractor should consider Common Weakness Enumeration (CWE), Software Engineering Institute (SEI) Top 10 secure coding practices and other sources when defining the standards [6].

2. ___ Does the SOW require the contractor to use static analysis tools to identify violations of the secure design and coding standards for critical function components?

3. ___ Does the SOW require design and code inspections to identify violations of secure design and coding standards for critical function components?

4. ___ Does the SOW require the mitigation of common software vulnerabilities? Derive from
   - Common Weakness Enumeration (CWE)
   - Common Vulnerabilities and Exposures (CVE)
   - Common Attack Pattern Enumeration and Classification (CAPEC)

5. ___ Does the SOW require penetration testing based upon malicious insertion and other security abuse cases?

6. ___ Does the SOW require specific code test-coverage metrics to ensure adequate testing of critical function components?

7. ___ Does the SOW require regression tests following changes to critical function code?

8. ___ Does the System Requirements Document require software fault detection, fault isolation (FDFI), and tracking (or logging) of faults and cybersecurity attacks?

9. ___ Does the SOW require critical function developmental software to be designed with least privilege to limit the number, size, and privileges of system elements?

10. ___ Does the System Requirements Document require a separation kernel or other isolation techniques for Level I critical function components to control communications between Level I critical functions and other critical and noncritical functions?

11. ___ Does the System Requirements Document require a software load key to encrypt and scramble software to reduce the likelihood of reverse engineering?

12. ___ Does the Systems Requirements Document require parameter checking and validation for the interfaces to critical function components?

13. ___ Does the SOW require that access to the development environment be controlled with limited authorities (least privilege), and does it ensure logging and tracing of all code changes to specific individuals?

14. ___ Does the SOW require commercial off-the-shelf (COTS) product updates to be applied and tested within a specified time period after release from the original equipment manufacturer or other software provider?

Questionnaire Instructions

For each critical function component or group of components, answer the questions covering supply chain vulnerabilities (part I) and software development vulnerabilities (part II).

Add domain-specific questions or any questions that the program developed relative to security threats.

Review each question and determine if the intent of the question applies to your acquisition. If it does not, mark it N/A. If it does, continue:

Determine whether your current vulnerability mitigation plans in the Statement of Work (SOW) or system requirements document address the question. If so, place a "Y" [Yes] in the blank. If not, place an "N" [No] in the blank.

Questions with a "No" response indicate areas in which the program should consider a countermeasure to mitigate risk.

One way of translating the "No" responses into risk likelihood is to map the percentage of "No" responses to a risk likelihood value, as shown in Table A-1.

Table A-1. Sample Risk Likelihood Mapping

Number of No Responses	Risk Likelihood
All No	Near Certainty (VH)
>=75% No	High Likelihood (H)
>= 25% No	Likely (M)
<= 25% No	Low Likelihood (L)
<= 10% No	Not Likely (NL)

Table A-2 provides sample summary of the vulnerability and threat assessment results used to develop the risk likelihood. A program could use such a table to clarify the rationale for the risk likelihood and should document the rationale in the Risk section of the PPP.

The overall risk likelihood is derived from the supply chain risk likelihood, the software assurance risk likelihood, and the threat assessment. The overall risk likelihood may be derived by using a weighted average of the three entries or using the highest risk. In the example shown in Table A-2, the overall risk likelihood of "High" was derived by applying equal weights for the supply chain and software assurance risk likelihood and the threat assessment risk. The program or Service may develop its own weightings based upon the program-specific and domain-specific knowledge.

Table A-2. Risk Likelihood Derived from Vulnerability and Threat Assessments

Critical Function Component	Mission Impact	Supply Chain Risk Likelihood	Software Assurance Risk Likelihood	Threat Assessment Risk	Overall Risk Likelihood
Component 1	I	High • No blind buys • No supply chain visibility • No supplier qualification process • No receiving verification No trusted suppliers	Very High • No fault logging • No secure design standard • No static analysis • No Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE), Common Attack Pattern Enumeration and Classification (CAPEC) • No input validation • No development environment controls • No regression test Low test coverage	Medium	High
Component 2	II	Low • No supply chain visibility No supplier qualification	Not Likely	Medium	Low

The "No" responses help the program determine the possible countermeasures to be considered for risk mitigation. A similar table may be created to record the countermeasures planned and the new risk probability as a result of the planned mitigations.

Appendix B: Vulnerability Database Assessment

A variety of weaknesses in software capabilities allow those capabilities to become exploitable by an attacker, thus allowing the attacker to influence, subvert, or otherwise make use of the critical software capabilities in ways that were never intended. If these exploitable weaknesses occur in a packaged piece of software provided commercially or through open source, it will often be assigned a Common Vulnerabilities and Exposures (CVE) identifier to help correlate the information and resources available about a particular vulnerability in software in use throughout the world.

The same types of mistakes and flaws that produce the vulnerabilities in commercial and open source capabilities also occur in noncommercial custom software developed for military end use. The Department of Defense (DoD) must ensure that these types of vulnerabilities are not present in critical mission systems. DoD is formalizing the methods and directives for software assurance.[1]

At the same time, a growing number of software developers and systems engineering practitioners possess the requisite training and experience to recognize, consider, and avoid these weaknesses, and a growing number of tools and techniques are available to review and test for the weaknesses through a variety of detection methods.

The following approach provides a way to use the information in the public vulnerability databases of CVE [1], Common Attack Pattern Enumeration and Classification (CAPEC) [2], and Common Weaknesses Enumeration (CWE) [3] to assess system vulnerabilities in a methodical way.

This method assumes that a set of secure design and coding standards have been established for custom developed or custom legacy software [5].

1. Determine the applicable attack vectors from CAPEC that will be used for the assessment. These are the vectors the systems engineer considers an attacker would use to try to gain access, control, or influence over a system once it is operational.
2. For each critical component, determine whether the component is a commercial off-the-shelf (COTS) product (including open source) or customer-developed product:
    - For COTS products, use the CVE database to identify a set of vulnerabilities associated with each CAPEC attack. In the DoD, the Information Assurance Vulnerability Alerts (IAVA) map to CVE [4], which provides a link to the several hundred commercial tools and services offering CVE capabilities that leverage CVE identifiers from more than 100 organizations in more than 25 countries.
    - For customer development software, use the CWE database to identify potential weaknesses associated with each attack. For each weakness, determine whether it is prohibited by the secure design and coding standard.

[1] Congress has included a definition of "Software Assurance" in Public Law 112-239 Section 933 in which software assurance is defined 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."

3. Determine the risk likelihood of the weakness or vulnerability using the following scale:

Near Certainty (VH)
High Likelihood (H)
Likely (M)
Low Likelihood (L)
Not Likely (NL)

   - For COTS products, assess the CVE database vulnerabilities as high or very high likelihood since the vulnerability is known and in the public domain. The likelihood may be adjusted if there is already mitigation in place that makes the vulnerability difficult to exploit or eliminates the vulnerability.
   - For weaknesses in custom-developed software that are prohibited by the secure design and coding standards, set the likelihood to high or very high. For other weaknesses, use the CWE description of technical impacts and knowledge of the design to determine whether the access to the weakness is already mitigated. Based upon that analysis, assign the likelihood.

4. For each weakness or vulnerability, identify possible mitigations. For weaknesses, the CWE database lists alternatives to mitigate the vulnerability. For CVE vulnerabilities, request mitigation or a fix from the supplier. If there is no supplier mitigation available, consider whether a mitigation needs to be added to the way the COTS products are used to make it difficult to exploit the known vulnerability.
5. Combine the likelihoods for each of the components. There are several ways to approach this. One is to simply average the vulnerabilities. Another is to take the highest likelihood. A third is to take the highest likelihood and scale it up based upon the number of vulnerabilities. This third method takes into account that the more vulnerabilities a component has, the more likely it is to be compromised.
6. Repeat these steps periodically to account for the elaboration of designs from high-level design to low-level design to code and for updates to the CVE database for COTS products. The assessment should be repeated before each Systems Engineering Technical Review (SETR) or when significant additional design detail has been developed. Timely mitigation of vulnerabilities requires reassessments and mitigations to keep risk at acceptable levels.
7. Use the vulnerability assessment results to inform the risk assessment and the risk-based cost-benefit trade-off.

Figure B-1 illustrates the evaluation of custom software for vulnerabilities. The attack patterns are identified (step 1) to represent the expected threat and identify the subset of weaknesses associated with the attack (step 2). These weaknesses can be used to influence the actions taken with a system design and architecture; to create security requirements; and to determine the likelihood the component or its supply chain will be compromised, and other aspects about how it will meet its mission support objectives.

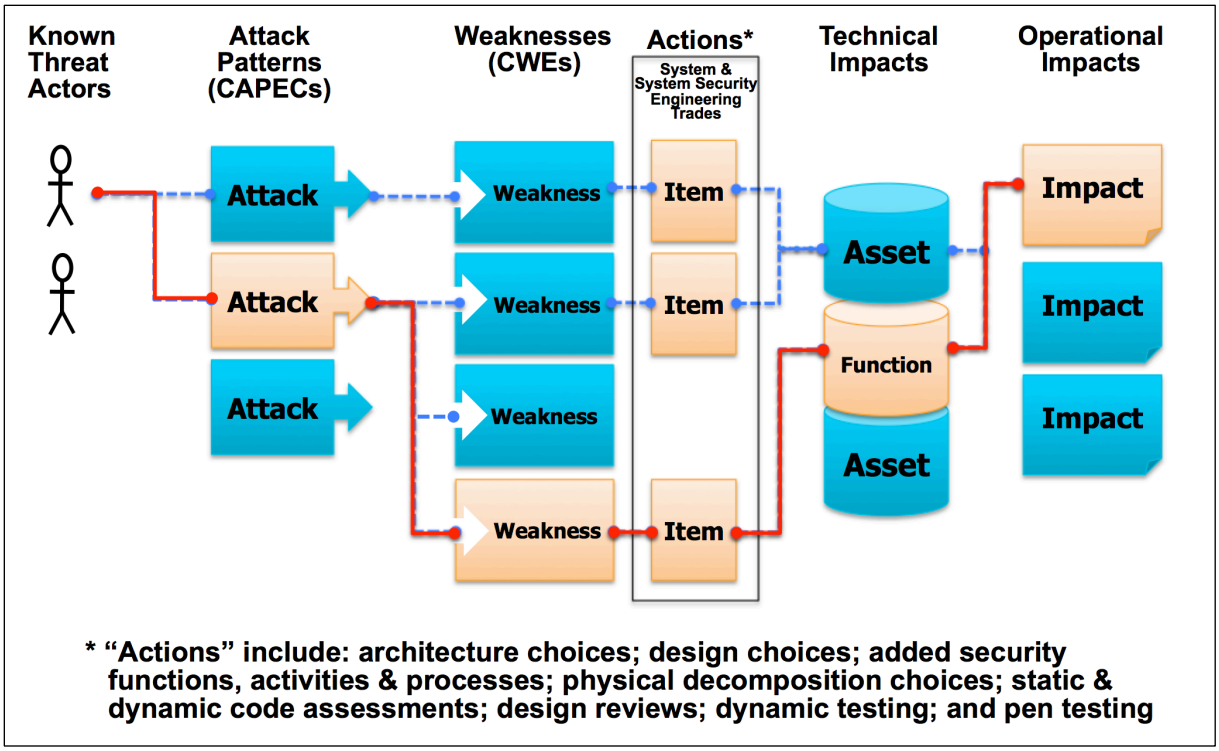

Figure B-1. Evaluation of Custom Software for Vulnerability

The description in this section draws upon the work described by Robert A. Martin [7].

References

[1] "The Common Attack Pattern Enumeration and Classification (CAPEC™) Initiative", MITRE Corporation, (https://capec.mitre.org/)

[2] "The Common Weakness Enumeration (CWE™) Initiative", MITRE Corporation, (https://cwe.mitre.org/)

[3] "The Common Vulnerabilities and Exposures (CVE®) Initiative", MITRE Corporation, (https://cve.mitre.org/)

[4] IAVM to CVE, Defense Information Systems Agency (DISA), (http://iase.disa.mil/stigs/iavm-cve.html).

[5] Fundamental Practices for Secure Software Development 2nd Edition: A Guide to the Most Effective Secure Development Practices in Use Today. Software Assurance Forum for Excellence in Code (SAFECode), (http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf)

[6] 2011 CWE/SANS Top 25 Most Dangerous Software Errors, MITRE Corporation, (http://cwe.mitre.org/top25/)

[7] Non-Malicious Taint: Bad Hygiene is as Dangerous to the Mission as Malicious Intent, CrossTalk Magazine issue on Mitigating Risks of Counterfeit and Tainted Components, March 2014, (http://www.crosstalkonline.org/storage/issue-archives/2014/201403/201403-Martin.pdf)

[8] Supply Chain Attack Framework and Attack Patterns, Dr. John F. Miller, MITRE Corporation, 2013.

Acronyms

CA	criticality analysis
CAPEC	Common Attack Pattern Enumeration and Classification
CDD	Capability Development Document
CDR	Critical Design Review
CI	configuration item
CI	counterintelligence
COTS	commercial off-the-shelf
CPD	Capability Production Document
CPI	critical program information
CTA	capstone threat assessment
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DAG	Defense Acquisition Guidebook
DASD(SE)	Deputy Assistant Secretary of Defense for Systems Engineering
DIA	Defense Intelligence Agency
DoD	Department of Defense
DODAF	Department of Defense Architecture Framework
DoDI	Department of Defense Instruction
FPGA	field-programmable gate array
FTA	fault tree analysis
IAVA	Information Assurance Vulnerability Alert
ICD	Initial Capabilities Document
IT	information technology
JCIDS	Joint Capabilities Integration and Development System
NIST	National Institute of Standards and Technology
PDR	Preliminary Design Review
PPP	Program Protection Plan
RFP	Request for Proposal
RMF	Risk Management Framework
SBS	system breakdown structure
SCRM	supply chain risk management

SETR	Systems Engineering Technical Review
SFR	System Functional Review
SME	subject matter expert
SOW	Statement of Work
SRR	System Requirements Review
SSE	system security engineering
TAC	Threat Assessment Center
TSN	trusted systems and networks
WBS	work breakdown structure

Trusted Systems and Networks (TSN) Analysis

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