Software Engineering Institute

Evaluating and Mitigating Software Supply Chain Security Risks

Robert J. Ellison
John B. Goodenough
Charles B. Weinstock
Carol Woody

May 2010

TECHNICAL NOTE
CMU/SEI-2010-TN-016

Research, Technology, and System Solutions (RTSS) and CERT Programs

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

List of Figures

List of Tables

Acknowledgements

We are grateful for the comments from those who reviewed a draft of this report: Nancy Mead, Julia Allen, Michele Moss, and Nadya Bartol.

Executive Summary

Managing supply chain risk is of widespread interest. Typically the focus is on manufacturing, where the goal is to minimize disruptions that would affect production or to ensure that low-quality or counterfeit products do not get incorporated into systems. Although software supply chain risk management has some of these aspects (e.g., a system may depend on the timely delivery of a subcontractor's product), the relative ease with which software can be modified changes the supply chain focus to (1) minimizing opportunities for unauthorized changes and (2) having appropriate methods for gaining confidence that such opportunities have been minimized, particularly by lower level participants in the supply chain. In addition, because software systems can be configured and used in ways that increase security risk, the end user of a software system has more responsibility to ensure against unauthorized product modification than is usually the case for end users of hardware systems. For software systems, the supply chain security risk management process must consider the potential introduction of security risks during deployment, configuration, and system operation, as well as during design and development.

For those who commission the development of systems (such as the Department of Defense [DoD]), limiting supply chain security risks initially means defining the security properties needed for the system being acquired, then evaluating and monitoring a supplier's ability to produce systems having these properties. Monitoring requires contractual language that gives the right to review certain information (such as the supplier's training and coding practices) as well as the technical capacity to conduct appropriate reviews. In addition, the acquiring organization must evaluate the supplier's approach to managing its software supply chain security risks—the risks it inherits from its suppliers.

The acquiring organization's job doesn't end with vetting a supplier: the delivered system must be configured appropriately when it is installed, and procedures must be in place to ensure that the system is operated in a secure manner (e.g., that newly discovered vulnerabilities are patched and that end-user adaptations don't introduce new vulnerabilities).

In this report, we identify software supply chain security risks that must be managed throughout the acquisition life cycle, and we specify the evidence that must be gathered to determine whether these risks have been appropriately mitigated. Evidence of supply chain security risk mitigation needs to be gathered at every phase of an acquisition's life cycle: initiation, development, configuration/deployment, operations/maintenance, and disposal. Required evidence includes analyses of a supplier's ability to produce secure software, security analyses of delivered products, evaluations to ensure control of access to the product at each step in the supply chain, and analyses of procedures for ensuring that the product is configured appropriately throughout its operational life.

This report provides an *assurance case* reference model showing how the gathered evidence is combined into an argument demonstrating that supply chain security risks have been addressed adequately throughout the acquisition life cycle. The reference model emphasizes two key strategies for controlling security risk: (1) identifying and monitoring a system's attack surface and (2) developing and maintaining a threat model. An implementation of these strategies requires differ-

ent actions at different phases of the acquisition life cycle, and this is reflected in the reference model.

The reference model is only an initial version; we expect changes to be made as we gain more experience with applying it to various programs and projects. Despite its preliminary state, we used it to guide our analysis of a specific DoD program to see to what extent appropriate evidence could be gathered and to what extent the evidence indicated that supply chain issues were being addressed adequately. Among other findings, we discovered that due to contractual limitations, much of the desired evidence could not be made available to the government without a contractual modification (and presumably, increased cost). In addition, the project was typical in its focus on limiting unauthorized access through infrastructure defenses such as firewalls, authentication protocols, and role-based access control. As is typical in our experience, little attention was being given to reducing the security impact of application code vulnerabilities, which are the major sources of security breaches in modern, web-enabled, systems today.

Abstract

The Department of Defense (DoD) is concerned that security vulnerabilities could be inserted into software that has been developed outside of the DoD's supervision or control. This report presents an initial analysis of how to evaluate and mitigate the risk that such unauthorized insertions have been made. The analysis is structured in terms of actions that should be taken in each phase of the DoD acquisition life cycle.

1 Introduction

1.1 The Software Supply Chain

1.1.1 Terminology

For the military, *supply chains* typically involve the movement of materials from home base to troops in theater. The responsibility for managing these supply chains falls to the acquisition and logistics experts. Traditionally, supply chains have also been linked to the movement of raw materials and subcomponents through a manufacturing process for consumer products such as automobiles and appliances. “A typical supply chain begins with ecological and biological regulation of natural resources, followed by the human extraction of raw material, and includes several production links (e.g., component construction, assembly, and merging) before moving on to several layers of storage facilities of ever-decreasing size and ever more remote geographical locations, and finally reaching the consumer” [Wikipedia 2009a].

Supply chain risk management usually[1] refers to limiting the risk of supply disruptions, typically disruptions that delay deliveries of an item to a manufacturer or consumer. The term can also refer to ensuring that inferior or counterfeit components are not introduced by suppliers. Both kinds of risks can occur in the software supply chain. For example, failure to produce or deliver a software component on time can delay delivery of a software system that depends on the component, and the delivery of faulty code or use of an inferior substitute software component can compromise the behavioral properties of the entire system in which the component is placed.

In this report, we focus on reducing the risk that an unauthorized party can change the behavior of software in a way that adversely affects its *security properties*. Security properties include confidentiality (preventing the unauthorized disclosure of information), integrity (preventing unauthorized changes to data), and availability (assurance that the capability provided by the software can be used when needed) [Wikipedia 2009c].

We use the term *system* when emphasizing visibility into the *internal* elements of software, including visibility of how the software is built and maintained. Typically, a system is custom built for an *acquirer* by a *contractor*, but systems may also be produced by internal elements of an organization. Because the software is custom built, the acquirer has (if desired) maximum visibility into its internal elements (e.g., its design or code) as well as the processes used to build and maintain it.

We use the term *product* when emphasizing the *external* characteristics and functions of software, usually with the implication that neither its internal elements nor the processes used to build and maintain it are readily available for examination. Typically, we consider a product to be software produced by a *vendor* for sale to a variety of customers (e.g., a *COTS* product) but we sometimes

[1] Wikipedia defines it this way: “Supply Chain Risk Management (SCRM) is a discipline of Risk Management which attempts to identify potential disruptions to continued manufacturing production and thereby commercial financial exposure” [Wikipedia 2009b].

refer to a completed system as a product when we mean to emphasize its properties and capabilities rather than its internals or how it was built.

In this report, a *supplier* is any organization that provides products or services needed when developing, distributing, operating, or maintaining software.

1.1.2 The Software Supply Chain Problem

As outsourcing and expanded use of commercial off-the-shelf (COTS) and open source software products increase and as end users exploit opportunities to reconfigure or make limited additions to deployed products and systems, supply chain security risk becomes a growing concern. Software is rarely defect-free, and many common defects[2] can be readily exploited by unauthorized parties to alter the security properties and functionality of the software for malicious intent. Such defects can be accidentally or intentionally inserted into the software at any point in its development or use, and subsequent acquirers and users have limited ways of finding and correcting these defects to avoid exploitation.

Participation in the software supply chain is global, and knowledge of who has touched each specific product or service may not be visible to others in the chain. Typically, an acquirer such as a Department of Defense (DoD) program office will only know about the participants directly connected to it in the supply chain and will have little insight into its suppliers' suppliers, as shown in Figure 1. Each of these indirect suppliers can insert defects for future exploitation.

Supply chain security risks must be addressed in every phase of the acquisition life cycle: initiation, development, configuration/deployment, operations/maintenance, and disposal. The view of the supply chain in Figure 1 applies primarily to the initiation and development phases of the acquisition life cycle. A somewhat different picture applies to the operations/maintenance phase, as outlined in Figure 2, where software supply chain security risk occurs through the delivery of sustainment upgrades and configuration changes. In addition, coding and design defects newly identified and reported as vulnerabilities may require patches and special security monitoring to prevent compromise, and these patches are delivered by various suppliers.

Software acquisition has grown from the delivery of stand-alone systems to the provisioning of capabilities integrated within a larger system-of-systems (SoS) context. This integration extends supply chain security risk. For example in the DoD, the Global Information Grid (GIG) interconnects all systems and software across the organization. Thus, the opportunity to introduce software security defects in a GIG product or service presents a supply chain security risk to every other member of the GIG.

[2] For example, "Improper Input Validation" is a common application software defect. It allows an attacker to present "input in a form that is not expected by the rest of the application. This … may result in altered control flow, arbitrary control of a resource, or arbitrary code execution" [MITRE 2009a]. Improper Input Validation is the highest ranked software error on the Common Weakness Enumeration's Top 25 list of the most dangerous programming errors [MITRE 2009b].

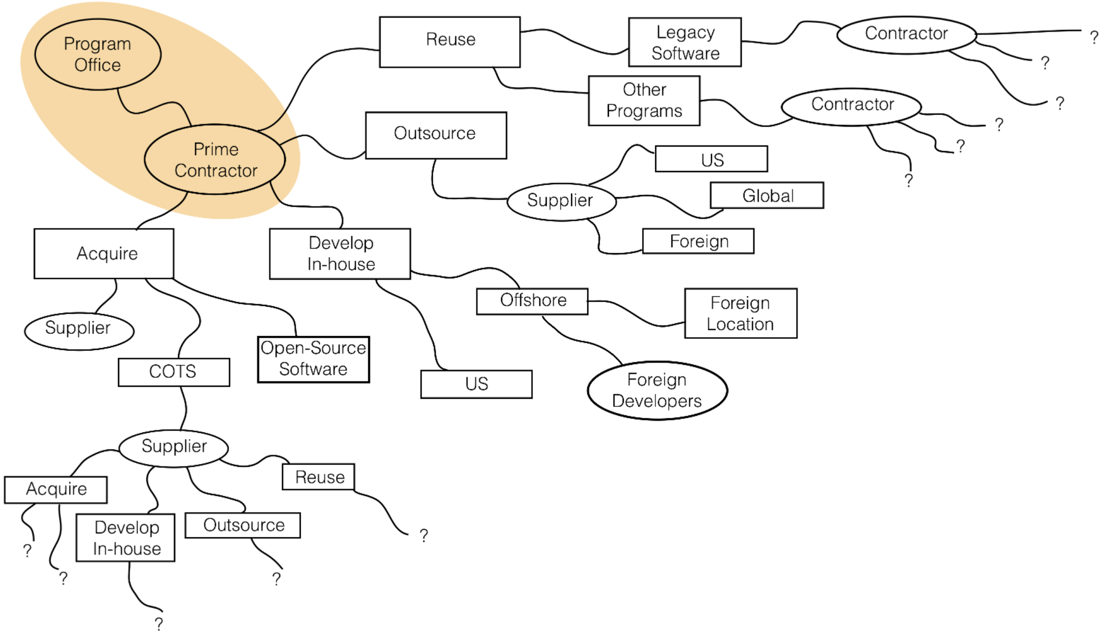

Figure 1: A Possible Supply Chain Relevant to the Initiation and Development Phases of the Acquisition Life Cycle

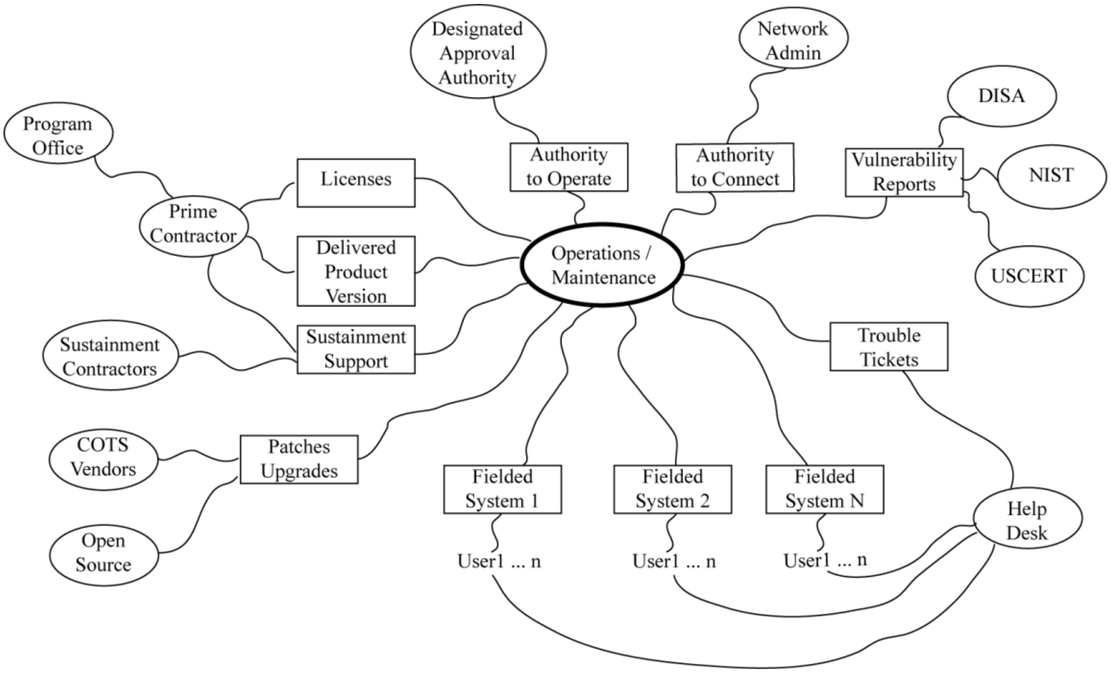

Figure 2: A Possible Supply Chain for the Operations/Maintenance Phase of the Acquisition Life Cycle

1.2 Software Supply Chain Security Risk

Of critical concern in today's highly interconnected software environment is the risk that an unauthorized party would change a product or system in ways that adversely affect its security properties. These software security risks are introduced into the supply chain in several ways:

- poor security requirements that lead to ineffective security considerations in all acquisition steps.
- coding and design defects incorporated during development that allow the introduction of code by unauthorized parties when the product or system is fielded. In addition, there are those defects that compromise security directly by allowing unauthorized access and execution of protected functionality.
- improper control of access to a product or system when it is transferred between organizations (failures in logistics), allowing the introduction of code by unauthorized parties.[3]
- insecure deployed configuration (e.g., a deployed configuration that uses default passwords).
- operational changes in the use of the fielded product or system that introduce security risks or configuration changes that allow security compromises (configuration control and patch management).
- mishandling of information during product or system disposal that compromises the security of current operations and future products or systems.

Software supply chain security risk exists at any point where organizations have direct or indirect access to the final product or system through their contributions as a supplier. Suppliers include distributors, transporters, and storage facilities, as well as organizations directly responsible for creating, enhancing, or changing product or system content. Without mitigation, these risks are inherited from each layer in the supply chain, increasing the likelihood of a security compromise.

Reduction of supply chain security risk requires paying attention to all of the following within the acquisition life cycle:

- **acquirer capabilities:** policies and practices for defining the required security properties of a particular product or system (not addressed in this initial version)
- **supplier capability:** ensuring that a supplier has good security development and management practices in place throughout the life cycle
- **product security:** assessing a completed product's potential for security compromises and determining critical risk mitigation requirements
- **product logistics:** the methods for delivering the product to its user and determining how these methods guard against the introduction of malware while in transit
- **operational product control:** ensuring that configuration and monitoring controls remain active as the product and its use evolve over time

[3] Focusing on controls to prevent supply chain tampering with a software product is a principle objective of the Software Supply Chain Integrity Framework being developed by SAFECode [Simpson 2009]. The framework pays particular attention to controls needed to ensure that products are moved securely along the supply chain (i.e., that customers receive the products a supplier intended to provide).

- **disposal:** ensuring software data and modules are effectively purged from hardware, locations, libraries, etc. when removal is needed (not covered in this initial version)

Addressing these risks impacts each phase in the acquisition life cycle and becomes a shared responsibility of the program office, each supplier, and operations management. Both the security of the supply chain and the security of the resulting product or system need to be considered. For each acquisition life-cycle phase, Table 1 identifies key activities that are needed in order to focus the proper attention on software supply chain security risk.

Table 1: Acquisition Life-Cycle Phases and Corresponding Supply Chain Security Risk Management Activities

Acquisition Phase	**Key Activities for Managing Software Supply Chain Security Risks**
Initiation	Perform an initial software supply chain security risk assessment and establish required security properties. Include supply chain security risk management as part of the RFP. Develop plans for monitoring suppliers. Select suppliers that address supply chain security risk.
Development	Monitor practices for supply chain security risk management. Maintain awareness of supplier's sub tier relationships.
Configuration/Deployment	Assess delivered products/systems. Configure/integrate with consideration of supply chain security risks. Develop user guidance to help mitigate supply chain security risk.
Operations/Maintenance	Manage security incidents. Review operational readiness. Monitor component/supplier.
Disposal	Mitigate risks of information disclosure during disposal.

1.3 Outline of this Report

Section 2 of this report introduces an approach for identifying software supply chain security risk and uses this approach to provide details of how the key risk management activities in Table 1 could be performed in different phases of the life cycle. Section 3 provides an initial version of an assurance case reference model covering three segments of the acquisition life cycle (development, configuration/deployment, and operations/maintenance). The model identifies what supply chain security risks need to be addressed, what evidence needs to be gathered to show that the risks have or have not been addressed adequately, and how the evidence supports claims that risks have been adequately addressed. Section 4 summarizes a review of a current program acquisition using the assurance case reference model presented in Section 3. The intent of this review was to exercise the reference model and to identify some of the ways in which current practice does and does not address software supply chain security risk. Section 5 summarizes the report and suggests future directions.

The analysis presented in this document is just an initial description of how to determine whether supply chain security risks are being adequately addressed. In particular, we focus on supplier capabilities to manage supply chain risks and don't address acquirer capabilities. We also don't address what supply chain risks need to be managed when a system is retired. Analysis of these aspects of software supply chain security risk should be included in future revisions of this report, and in particular, in revisions of the reference model presented in Section 3.

2 Supply Chain Security Risk Analysis and Mitigations

Software vulnerabilities, in general, are a major contributor to software security risk. It is impossible, as well as impractical, to eliminate all software vulnerabilities, many of which can lead to supply chain security risk. However, there are key strategies for reducing and managing such risks. Our discussion is based on current thinking and emerging practices that are generally considered to be most effective in managing security risks arising from the nature of software and its role in the software supply chain.

Two powerful strategies—one focused on understanding and controlling a system's *attack surface*[4] and the other focused on understanding potential threats (*threat modeling*)—are key to making supply chain security risk management tractable. A system's attack surface characterizes potential vectors for compromising a system. Threat modeling characterizes which aspects of the attack surface are most at risk for exploitation.[5] These concepts are useful during the development, deployment, and operations/maintenance of a system. They help guide what information must be gathered and how it can be best used to help prioritize and mitigate (if not eliminate) supply chain security risks.

Software security risks addressed by threat modeling and attack surface analysis are quite different from those that infrastructure security mechanisms (such as security regulations and information assurance processes) tend to address. Infrastructure mechanisms prevent unauthorized individuals from gaining access to system code and data. Such mechanisms include the appropriate deployment of firewalls, the use of strong passwords for authentication, and authorization mechanisms such as role-based access control. These can be effective against certain classes of threats, but as systems become more dynamically connected and as operating system and network security vulnerabilities are reduced, application software itself becomes the next attack target.

Security risks in application code have been ignored in part because of the faulty assumption that firewalls and other perimeter defenses will deny access to those with malicious intent. However, these defenses are insufficient as the user base for DoD applications grows into the tens of thousands and includes DoD organizations with different security sensitivities as well as coalition partners. In such environments, application code itself is vulnerable to attack and thus becomes a source of software security risk. This risk is compounded because application developers typically have not been trained in how to develop secure software and so are unaware of the security risks that can be introduced during application design and coding.

Mitigation of software supply chain security risk requires that more attention be given to application software security. Today, over 25 large-scale application software security initiatives are underway in organizations as diverse as multi-national banks, independent software vendors, the U.S. Air Force, and embedded systems manufacturers. The Software Assurance Forum for Excel-

[4] The notion of an attack surface was originally defined by Howard and Lipner [Howard 2003]. See Section 2.2.

[5] Threat modeling is part of Microsoft's Secure Development Lifecycle [Swiderski 2004, Howard 2006]. For more information, go to http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx.

lence in Code (SAFECode), an industry-led non-profit organization that focuses on the advancement of effective software assurance methods, published a report on secure software development [Simpson 2008]. In 2009, the first version of *The Building Security in Maturity Model* (BSIMM) was published [McGraw 2009].[6] The Software Assurance Processes and Practices working group,[7] operating under the sponsorship of the Department of Homeland Security's National Cyber Security Division, has released several relevant documents, including a *Process Reference Model for Assurance* linked to the CMMI-DEV model [SAPPWG 2008]. In addition, the Open Web Applications Security Project (OWASP) has developed a Software Assurance Maturity Model (SAMM) for software security [OWASP 2009]. Finally, the build-security-in website[8] contains a growing set of reference materials on software security practices.

Increased attention on secure application software components has influenced security testing practices. All of the organizations contributing to the BSIMM do penetration testing,[9] but there is increasing use of fuzz testing. Fuzz testing creates malformed data and observes application behavior when such data is consumed. An unexpected application failure due to malformed input is a reliability bug and possibly a security bug. Fuzz testing has been used effectively by attackers to find vulnerabilities. For example, in 2009, a fuzz testing tool generated XML-formatted data that revealed an exploitable defect in widely used XML libraries [Codenomicon 2009]. At Microsoft, about 20 to 25 percent of security bugs in code not subject to secure coding practices are found via fuzz testing [Howard 2006].

2.1 Acquisition and Operational Contexts

Supply chain security risks and their mitigations depend on the kind of acquisition and the phase of the acquisition life cycle. Acquisitions can include the purchase of commercially available software, development of custom software, an enhancement to an existing system, or maintenance activities such as patching software defects in existing systems. For a custom-developed system, an acquirer has the opportunity to specify security properties the system is required to satisfy (e.g., what information is to be protected, what availability is expected, and what degree of protection is desired). In this kind of acquisition, the acquirer can also impose requirements for *evidence* that the required security properties will hold, such as

- architecture and design analyses
- information on development coding practices
- the existence of an RFP requirement to provide an attack surface analysis and mitigation plan
- plans to include security testing in acceptance tests
- the results of security tests

[6] BSIMM was created from a survey of nine organizations with active software security initiatives that the authors considered to be the most advanced. The nine organizations were drawn from three verticals: financial services (4), independent software vendors (3), and technology firms (2). Those companies among the nine who agreed to be identified include: Adobe, The Depository Trust & Clearing Corporation (DTCC), EMC, Google, Microsoft, QUALCOMM, and Wells Fargo.

[7] See https://buildsecurityin.us-cert.gov/swa/procwg.html.

[8] See https://buildsecurityin.us-cert.gov/daisy/bsi/home.html.

[9] Penetration tests attempt to break into or compromise systems.

Of course, the acquirer of a commercially available product can only indirectly evaluate a supplier's ability to produce a secure software product, but the acquirer can test the product's security properties.

Analysis of supply chain security risks and mitigations goes beyond product and supplier assessments and has to include deployment and use. A typical software product provides more functionality than is required, and an attacker may be able to exploit those unused features. The required use also affects risks and mitigations. For example, a product feature that requires the processing of JavaScript or XML-formatted input expands the scope of the analysis of supply chain security risks and mitigations. Products are typically selected for their functionality (and not for their security properties), so a fully functional product may have inherent supply chain security risks that have to be mitigated during deployment.

2.2 Concentrate the Analysis—an Attack Surface

In 2003, Howard observed that attacks on Windows systems typically exploited a short list of features including [Howard 2003a]:

- open ports, services running by default, and services running with system-level privileges
- dynamically generated web pages
- enabled accounts, including those in administrative groups
- enabled guest accounts, weak access controls

Instead of considering the whole system, Howard proposed that analysis concentrate on the features that were most likely to be exploited. These features compose the system's *attack surface*. A system with a greater number of exploitable features has a larger attack surface and is at greater risk of exploitation. Howard's initially intuitive description of an attack surface led to a more formal definition with the following dimensions [Howard 2003b]:

- targets: data resources or processes desired by attackers (a target could be a web browser, web server, firewall, mail client, database server, etc.)
- enablers: processes and data resources used by attackers to reach a target (e.g., web services, a mail client, XML, JavaScript, or ActiveX[10])
- channels and protocols (inputs and outputs): used by attackers to obtain control over targets
- access rights: constraints intended to limit the set of actions that can be taken with respect to data items or functionality

As an example of the relation between attack surface analysis and software supply chain risk, consider that an increasing number of applications are XML-enabled, and XML is an attack surface enabler. The Finnish firm Codenomicon reported in August 2009 that vulnerabilities existed in XML libraries from Sun, the Apache Software Foundation, and the Python Software Foundation [Codenomicon 2009]. These vulnerabilities could result in successful denial-of-service attacks on any applications built with them. This is an example of software supply chain security

[10] Mechanisms such as JavaScript or ActiveX give the attackers a way to execute their own code.

risk, since the applications built using these libraries now have security vulnerabilities created not by the application builder but by the vendor of the XML product used in building the application.

An attack surface analysis reduces supply chain security risk in several ways:

- A system with more targets, more enablers, more channels, or more generous access rights provides more opportunities to the attacker. An acquisition process designed to mitigate supply chain security risks should include requirements for a reduced and documented attack surface.
- The use of product features influences the attack surface for that acquirer. The attack surface can define the opportunities for attacks when usage changes.
- Attack surface analysis helps to focus attention on the code that is of greatest concern for security risk. If the code is well- partitioned so that features are isolated, reducing the attack surface can also reduce the code that has to be evaluated for threats and vulnerabilities.
- For each element of a documented attack surface, known weaknesses and attack patterns[11] can be used to mitigate the risks.
- The attack surface supports deployment, as it helps to identify the attack opportunities that could require additional mitigation beyond that provided by the product.

2.2.1 The Use of Attack Surface Analysis in Different Acquisition Phases

To illustrate the way it can reduce supply chain security risk, let's consider how attack surface analysis could be incorporated into the phases of an acquisition. (The phases and activities are the same as those listed in Table 1 on page 5.)

2.2.1.1 Initiation

Perform an initial supply chain security risk assessment

This activity should identify—from the acquirer's perspective—critical aspects of the system that could be affected by supply chain security risks such as:

- Criticality of use and information. Criticality can be a factor in attack objectives. A data store with critical information could be a target. Use of critical functions or services could be targeted to disrupt operations.
- Known supply chain risks associated with technologies or design requirements. Newer technologies, such as web services or design patterns including service-oriented architectures (SOAs), have a short history of known attack patterns and a relatively short list of known coding and design weaknesses compared to more mature technologies, so they may present greater risks that should be addressed in the request for proposal (RFP).

[11] Attack patterns are "descriptions of common methods for exploiting software." [https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/attack.html] More information on attack patterns is available at the referenced URL as well as in Chapter 2 of *Software Security Engineering: A Guide for Project Managers* [Allen 2008].

Include supply chain security risk management as part of the RFP

- Require preliminary identification of attack surface elements. The proposal should discuss the risks of proposed technologies (enablers), the likely targets of an attack, and the system inputs and outputs (the channels of communications). An RFP responder could be asked to
  - □ build attack patterns and abuse cases that reflect potential threats
  - □ create technology-specific attack patterns
  - □ document and update the system and software attack surface

  It is difficult to verify an attack surface for a third-party product or whether attack surface knowledge has been used effectively in developing such products, because design and development artifacts are typically not made available to product users. Consequently, an acquisition for custom development should pay special attention to the contribution third-party components make to the attack surface.
- Require a detailed inventory of all planned third-party software and the role of each within the delivered solution.
- For third-party software that contributes to attack opportunities, an RFP responder should be asked to describe
  - □ security-related selection criteria and review processes. The selection criteria for COTS, government off-the-shelf (GOTS), and open source software can include historical evidence, independent assessments, and knowledge of the applicable development processes used.
  - □ how third-party software risks are mitigated.
  - □ the security testing done for third-party software.

2.2.1.2 Development

Monitor practices for supply chain security risk management

- Testing should target the attack opportunities identified by the attack surface.
- For custom software development, maintain the attack surface documented by the developer.
- For third-party software, maintain the attack surface assessments of third-party software and suppliers. Costs can be reduced if a central DoD organization identifies the third-party software attack surfaces and makes its analysis available to programs using this software.

2.2.1.3 Configuration/Deployment

Assess delivered products/systems

- The attack surface can guide acceptance testing. Penetration testing informally identifies attack surface components. An attack surface may be a contract deliverable, but the assessment should consider developing one independently, particularly for commercial products. Data input points in the code are good targets for fuzz testing.
- Confirm the detailed inventory of all planned third-party software and the role of each within the delivered solution.

Configure/integrate with consideration of supply chain security risks

- Configure the system to reduce the attack surface to one that is consistent just with user requirements and the available risk mitigations.

The acquirer should generate an attack surface based on actual use and operational risks. An analysis of the resulting attack surface can lead to additional installation-specific mitigations to reduce specific risks.

Develop user guidance to help mitigate supply chain security risk

- The site system administrative and development staff should document and communicate changes to the attack surface if currently unused features are enabled later.

2.2.1.4 Operations/Maintenance

Monitor component/supplier

- Review new releases for changes in the attack surface that can trigger needs for new mitigations.

2.3 Assess the Security Risk—Threat Modeling

The creation of an attack surface helps to focus analysis but does not identify security risks and possible mitigations for the application software. *Threat modeling* and other similar risk assessment techniques are a systematic approach to identify security risks in the software and rank them based on level of concern. Threat modeling constructs a high-level application model (e.g., data flow diagrams), a list of assets that require protection, and a list of risks. For example, a detailed walk-through of a data flow should

- consider the deployed configuration and expected use
- identify external dependencies such as required services
- analyze the interfaces to other components (inputs and outputs)
- document security assumptions and trust boundaries (e.g., a security control point)

As Steingruebl and Peterson state, undocumented assumptions have caused significant system failures [Steingruebl 2009]. The problem often occurs when systems with incompatible assumptions are composed. For example, a classic security problem with legacy systems is that they were often designed under the assumption of operating in a trusted and isolated operating environment. That assumption will be violated if such legacy systems are used as a back end for a web-based, public-facing, front end. Modeling the threat using a multi-system data flow could identify such assumption mismatches.

Consider the database usage example shown in Figure 3. Threat modeling would generate a user scenario for submitting a query. In this case, assume a scenario where the user submits an employee ID and receives in return the name and salary of that employee. The external dependencies include the database server. Security assumptions for the database server are that it maintains authentication information, encrypts sensitive data, and maintains access logs.

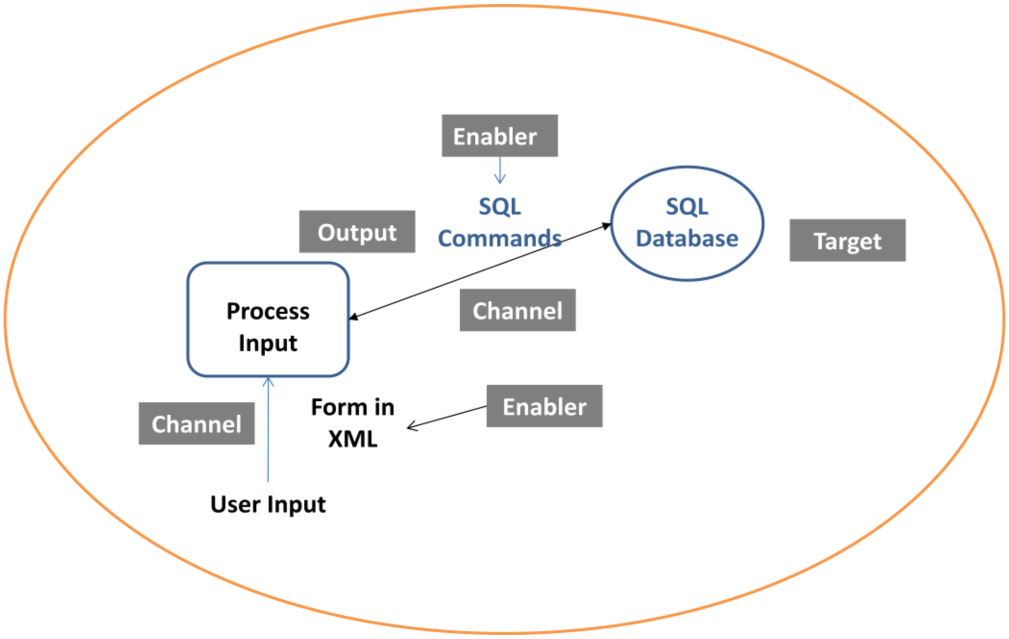

Figure 3: Attack Surface Example

Threat modeling then analyzes the data flow diagram as shown in Figure 3. Assume the user submits the ID value of *48983*. What data is sent from the application code to the database server? Input to the server should be a database command such as

SELECT ID name salary FROM EMPLOYEES WHERE ID=48983

The server then returns [48983 Sally Middleton $74,210].

Given properly formatted input, the data flow will successfully execute and return the desired data or an error if the ID is not found. Threat modeling analysis considers what could go wrong if the input is *not* formatted properly. In such a case, can an attacker induce improper behavior? In this case, the answer could be yes.

Suppose the user entered *ID=48983* | *1= 1* (where | means logical "or"). If the application code does not verify that the ID is a string of digits, the generated database command will now be

SELECT ID name salary FROM Employees WHERE ID=48983 | 1 =1

The database server interprets this command to mean *select entries where the ID is 48983* ***or*** *1 equals 1*. Since 1 is always equal to 1, the selection criterion is always true, and salary information is returned for all employees. The risk associated with this defect is high; variants of it have been used in attacks that caused credit card data to be downloaded illegally.

A risk assessment is never complete and cannot guarantee that application code is free of security-related defects. It is based on current knowledge. New attack techniques may exploit a defect in code that was previously determined to be secure.

The benefits of threat modeling or an equivalent risk analysis technique include the following [Howard 2006]:

- Identifies system risks before a commitment to a detailed implementation.
- Requires development staff to review the application software architecture and design from a security perspective.
- Supports the selection of countermeasures for the planned usage and operating environment as defined by user scenarios, dependencies, and security assumptions.
- Contributes to the reduction of the attack surface. There are some general ways to reduce an attack surface. The most obvious one is to review the importance of each feature. The detailed analysis associated with threat modeling can suggest other mitigations. For example, risk can be reduced by limiting access to the available functionality (e.g., by ensuring that code is executed with the minimum privileges needed to accomplish a task).
- Provides guidance for code review. The threat model developed for the system identifies the components that require an in-depth review and the kinds of defects that might appear. There are general design patterns such as using a canonical output format that can eliminate some of the potential defects.
- Guides security and penetration testing. Security and penetration testing should incorporate techniques that have been used successfully by attackers. There are commercial tools designed to support such a test approach.

2.3.1 The Use of Risk Assessments in Different Acquisition Phases

There are several ways of assessing security risk [NIST 2002, Haimes 2008, Howard 2006]. We provide a preliminary description of methods that are particularly useful to controlling software supply chain risk at the different phases of an acquisition life cycle. We focus particularly on the risks associated with the development and use of application code (i.e., code that interfaces directly with a user).

2.3.1.1 Initiation

Write software supply chain security risk management parts of RFP

The objective here is to require systematic analysis of the security risks associated with application code. A risk assessment should be a requirement for the initial development of a system, but many software acquisitions serve to integrate a new or revised component into an existing operational system. In this case, the risk assessment should focus on the resulting integrated system rather than the component alone. For example, the assessment should consider whether security is maintained by the integration and whether there are undocumented mismatches among the assumptions associated with the added component and the existing operational environment.[12] In any event, a documented risk assessment is needed to provide a baseline for custom development and for the integration of COTS, GOTS, or open source components.

[12] The likelihood of such mismatches increases with the use of COTS or GOTS software components.

Select suppliers that address supply chain security risk

The successful use of a risk assessment technique such as threat modeling depends on the knowledge, skill, and ability of the staff and the collective experience they can draw on. This criterion is reflected in the BSIMM study. One objective of that effort was to enable an organization to perform a self-assessment by comparing their practices to the collective practices of nine organizations. BSIMM identified 110 activities and created a framework to categorize them. The major categories are Governance, Software Security Knowledge,[13] Security Software Development Life-Cycle Artifacts,[14] and Deployment Practices. Using this framework, an acquisition could determine the information RFP responders should provide to show that they are prepared to deal with software supply chain security risk, such as

- Require RFP responders to evaluate their ability to address supply chain risk by considering the following areas (drawn from the BSIMM framework):
  - continuing developer staff training that includes
    - integrity of development—access controls and configuration management
    - awareness of supply chain security risks
    - design and coding practices that support the development of secure software
  - corporate knowledge
    - attack models: threat modeling, abuse cases, and technology-specific attack patterns
    - security features and design: usable security patterns for major security controls
    - standards and requirements: for authentication, authorization, input validation, etc.; security standards for technologies in use; and standards for third-party software use
- Require RFP responders to consider the capabilities of their suppliers to apply appropriate software security risk assessment and mitigation techniques. Does each supplier have access to staff knowledgeable about the threats and mitigations associated with the requested development? Is each supplier training its developers on recommended development practices needed for the final product? Does the responder share software supply chain security risks and mitigations with its suppliers, and do they share this information with *their* subcontractors? Do suppliers and their subcontractors provide insight into their risk assessment results?

Suppliers using security-focused development practices reduce supply chain security risk. These practices focus on architecture analysis, code reviews, and security testing. Microsoft's Security Development Lifecyle [Howard 2006] includes a full spectrum of security-focused application development practices. RFP language can encourage the use of these practices throughout the supply chain to reduce security risk.

2.3.1.2 Development

Monitor supply chain security risk management practices

- Ensure the incorporation of attack and vulnerability knowledge gained from the risk assessment into the development of application code.

[13] This is called "Intelligence" in the BSIMM.

[14] These are called "SSDL Touchpoints" in the BSIMM.

- Ensure consideration of security risk assessment results in architecture design and development.
- Ensure consideration of security risk assessment results in testing third-party supplied software.
- Identify risks mitigated by the following:
  - architecture mitigations for application code defects
  - architecture mitigations for supply chain security risks associated with third-party-developed software (subcontractors, COTS/GOTS, open source)
- Evaluate fuzz testing results using a system-specific risk assessment
- Evaluate the effectiveness of security testing applied to third-party-developed software
- Evaluate design and code reviews explicitly targeting risks identified in the risk assessment
- Evaluate effectiveness of the automated support for source code analysis (e.g., static analysis tools) based on a risk assessment

2.3.1.3 Configuration/Deployment

Assess delivered products/systems

- The risk assessment can guide acceptance testing areas, as was suggested for attack surfaces in Section 2.2.1.3.

Configure/integrate with consideration to supply chain security risks

- Risk assessment (e.g., threat modeling) should be done as part of the integration of the acquired software with the existing operational system. Are there mismatches among design, development, or operational assumptions that could be exploited? If a risk assessment was done for the existing system, that analysis should be updated and documented to reflect the changes.
- Acceptance of the product or system should include a review of the provided risk assessment to confirm its sufficiency and accuracy.

2.3.1.4 Operations/Maintenance

Monitor component/supplier

- Risks change as attackers develop new methods. A system risk assessment reflects the risks known at the time it was developed so it should be reviewed and updated periodically. Such a review should be part of the operating practices of both the product developer and user organization.

2.4 Summary and Final Note

Attack surface analysis and risk assessments such as threat modeling are key techniques for reducing software supply chain security risk. However, these analyses cannot be considered static artifacts. Attackers can introduce new techniques that are applicable to software that was thought to be secure. The attack surface and associated threat models need to be reviewed periodically. The reviews need to be more frequent for new technologies because these technologies have a relatively short history of exploits that are available to guide threat modeling and other security risk assessment techniques.

3 An Assurance Case Reference Model for Software Supply Chain Security Risk

3.1 Introduction

An assurance case is used to argue that available evidence supports a given claim. In this report, the claim of interest is that software supply chain security risk has been reduced, and the evidence is the collection of information relevant to showing that certain risk mitigations have been implemented effectively. The case shown here is our initial attempt to integrate the techniques and concepts discussed in Section 2.

The specific technique we use is the goal-structured assurance case. The case requires a goal (or claim) such as “The system is secure,” a set of evidence such as test results, and a detailed argument linking the evidence to the claim. Without an argument as to why the test results support the claim of security, an interested party could have difficulty seeing its relevance or sufficiency. With only a detailed argument depending on test results to show that a system was secure, but in the absence of those results, again, it would be difficult to establish the system’s security.

In our case, the top-level claim is “Supply chain security risks for product <P> have been reduced ALARP[15] (as low as is reasonably practicable).” From that claim flows an argument that supports the top-level claim. The argument consists of one or more subsidiary claims that, taken together, make the top-level claim believable. These lower level claims are themselves supported by additional claims until finally a subclaim is to be believed because evidence exists that clearly shows it to be true.

To develop the assurance case and make it reviewable by others, we use the Goal Structuring Notation (GSN) developed by Tim Kelly and his colleagues at the University of York in the United Kingdom [Kelly 1998]. This notation is ideally suited for explaining what information is needed and why it is needed to show that software supply chain security risks have been adequately addressed. Figure 4 shows a short assurance case documented in GSN. In it, the top-level claim is labeled C1. The argument consists of the subclaims, C2, C3, and C4. Claims C2 and C4 are supported by evidence, Ev1, Ev2, and Ev3.

Claims are phrased as predicates; they are either true or false. Evidence is stated as noun phrases. Other notations shown in the sample are

- the diamond under node C3, indicating that the claim requires further development
- the triangle under node Ev3, indicating that the evidence is parameterized and needs to be instantiated in an actual case
- the parallelogram, labeled S1, which explains how the argument is structured. (S stands for *strategy*.)

[15] ALARP is a principle used in safety cases. See http://en.wikipedia.org/wiki/ALARP.

- a rounded rectangle labeled Cx1; this node provides additional *contextual* information about the node to which it is attached
- an oval with an A under it labeled *Assumption*; this node is used to state assumptions not being further addressed by the case

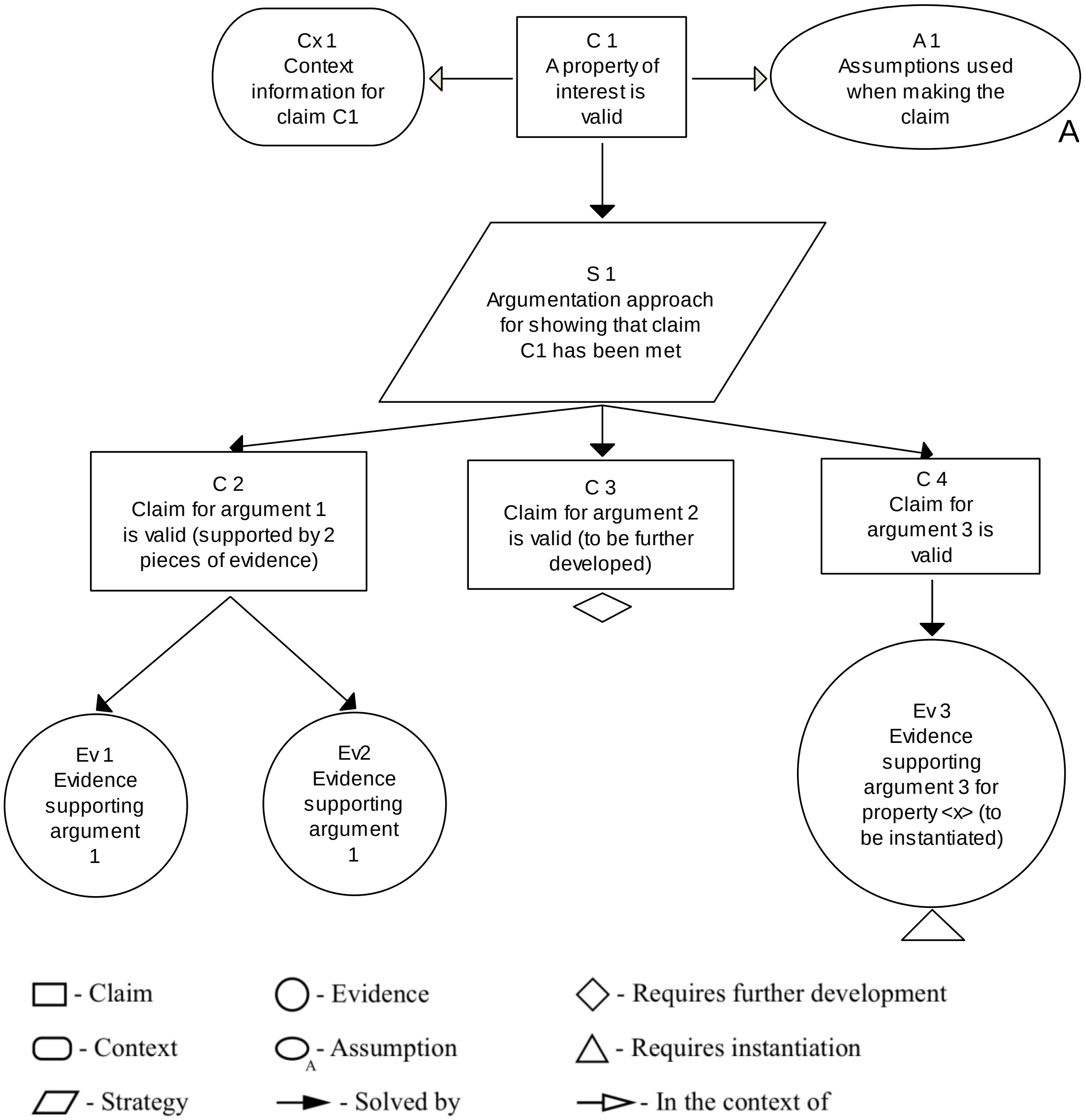
Cx 1
Context information for claim C1
C 1
A property of interest is valid
A 1
Assumptions used when making the claim
A
S 1
Argumentation approach for showing that claim C1 has been met
C 2
Claim for argument 1 is valid (supported by 2 pieces of evidence)
C 3
Claim for argument 2 is valid (to be further developed)
C 4
Claim for argument 3 is valid
Ev 1
Evidence supporting argument 1
Ev2
Evidence supporting argument 1
Ev 3
Evidence supporting argument 3 for property <x> (to be instantiated)
- Claim
- Context
- Strategy
- Evidence
- Assumption
A
- Solved by
- Requires further development
- Requires instantiation
- In the context of

Figure 4: Assurance Case Structure

The assurance case for supply chain security risk reduction (see Figure 5) starts with the claim "Supply chain security risks for product <P> have been reduced ALARP." To show that this claim is true, the argument addresses four largely independent concerns introduced in Section 1.2:

- supplier capability: ensuring that a supplier has good security development and management practices in place throughout the life cycle (addressed in claim C1.1)

- product security: assessing a delivered product’s potential for security compromises and determining critical risk mitigation requirements (addressed in claim C1.2)
- product logistics: the methods used to deliver the product to its user and how these methods guard against the introduction of malware while in transit (addressed in claim C1.3)
- operational product control: ensuring that the appropriate configuration and monitoring controls remain in place as the product and use evolve over time (addressed in claim C1.4)

The start of the assurance case shown below reflects these four concerns. We claim that if all four concerns are addressed satisfactorily, the claim of supply chain security risk reduction is valid.

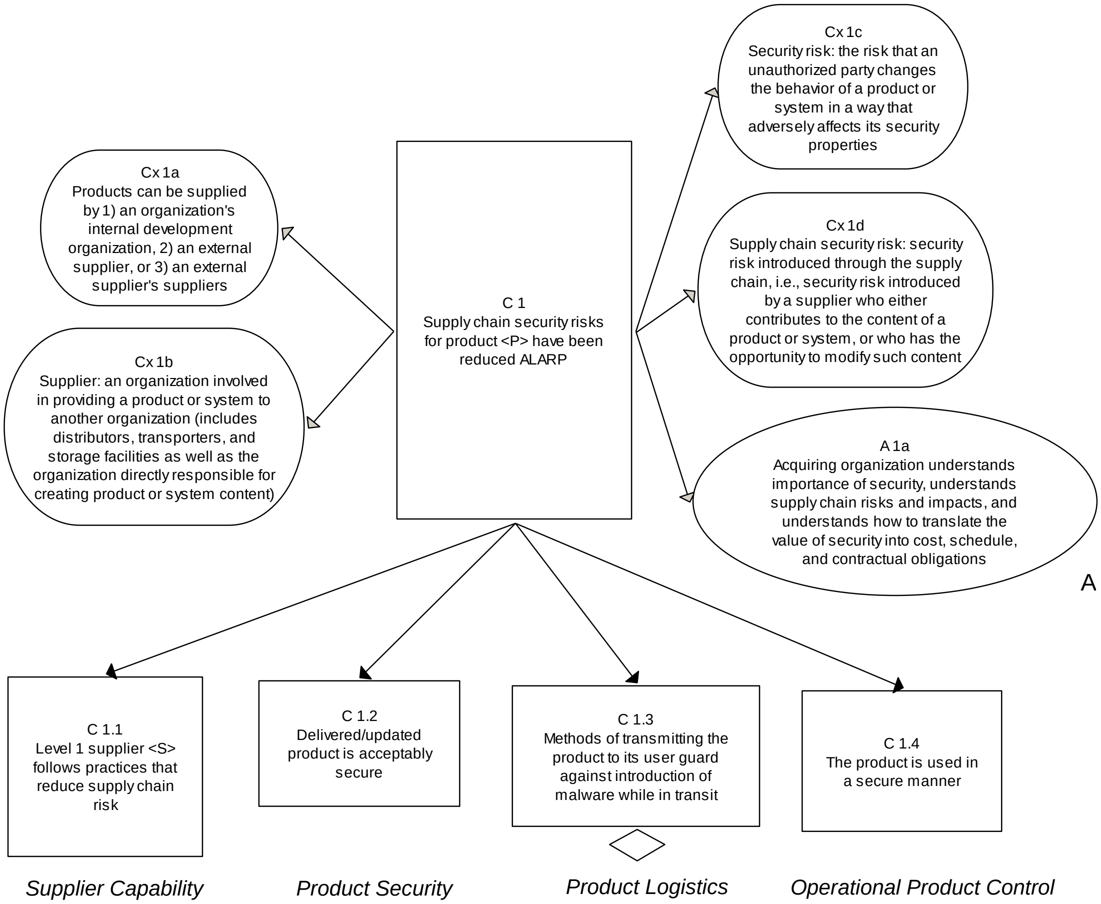
Cx 1c
Security risk: the risk that an unauthorized party changes the behavior of a product or system in a way that adversely affects its security properties
Cx 1a
Products can be supplied by 1) an organization's internal development organization, 2) an external supplier, or 3) an external supplier's suppliers
Cx 1d
Supply chain security risk: security risk introduced through the supply chain, i.e., security risk introduced by a supplier who either contributes to the content of a product or system, or who has the opportunity to modify such content
C 1
Supply chain security risks for product <P> have been reduced ALARP
Cx 1b
Supplier: an organization involved in providing a product or system to another organization (includes distributors, transporters, and storage facilities as well as the organization directly responsible for creating product or system content)
A 1a
Acquiring organization understands importance of security, understands supply chain risks and impacts, and understands how to translate the value of security into cost, schedule, and contractual obligations
A
C 1.1
Level 1 supplier <S> follows practices that reduce supply chain risk
C 1.2
Delivered/updated product is acceptably secure
C 1.3
Methods of transmitting the product to its user guard against introduction of malware while in transit
C 1.4
The product is used in a secure manner
Supplier Capability
Product Security
Product Logistics
Operational Product Control

Figure 5: Top-Level Assurance Case

3.2 The Level 1 Supplier Follows Practices that Reduce Supply Chain Security Risk

Figure 6 expands claim C1.1 stating that Level 1 supplier <S> follows practices that reduce supply chain security risk. We argue that the claim is valid if the supplier uses

- acceptable governance policies and practices that support security (addressed in claim C1.1.1.1)

- effective processes supporting the development of secure products (addressed in claim C1.1.1.2)
- good practices for responding to customer security problems (addressed in claim C1.1.1.3)
- good policies and practices for evaluating the level of supply chain security risk introduced by relying on *its* suppliers (addressed in claim C1.1.1.4)

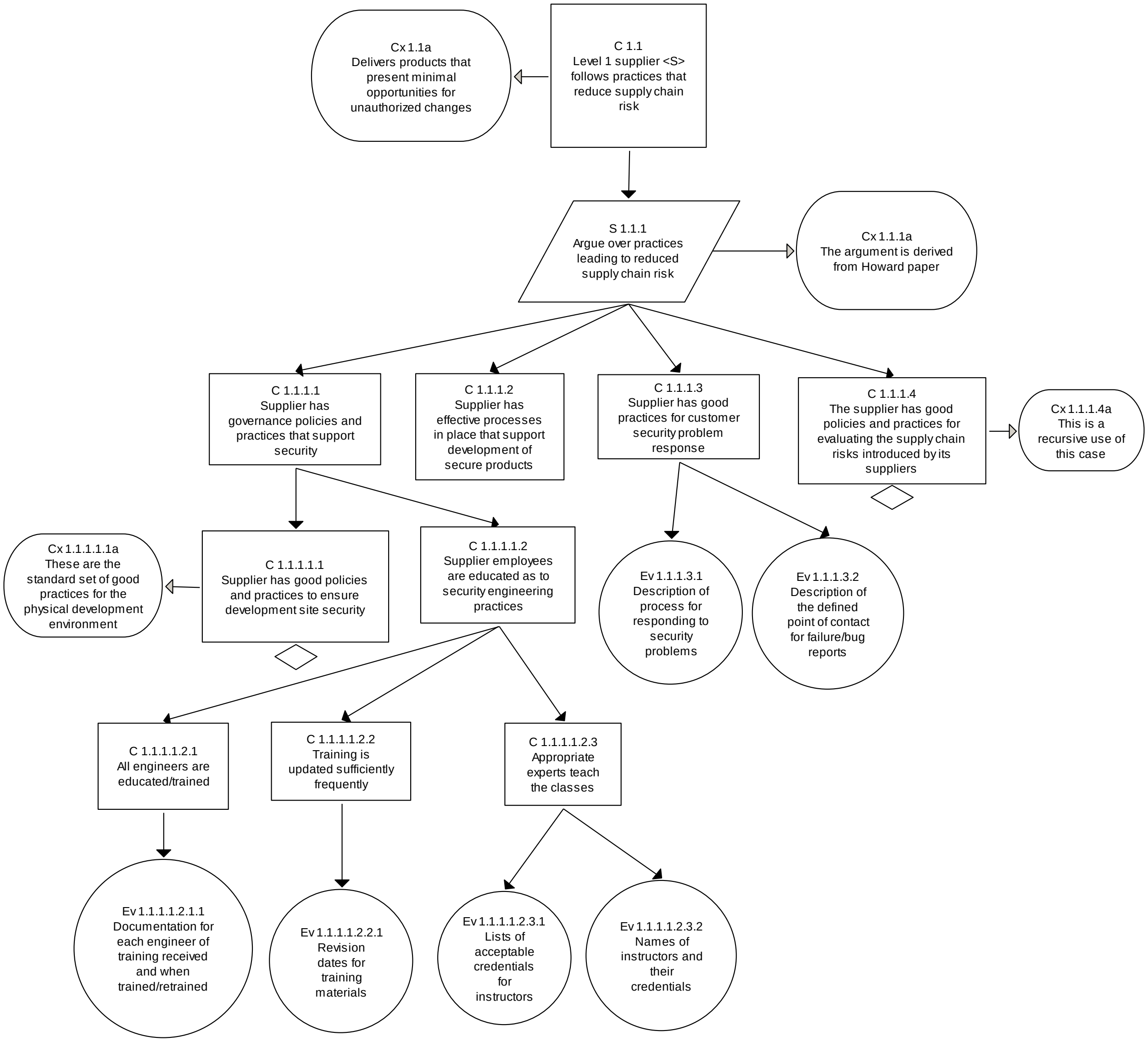
Cx 1.1a Delivers products that present minimal opportunities for unauthorized changes
C 1.1 Level 1 supplier <S> follows practices that reduce supply chain risk
S 1.1.1 Argue over practices leading to reduced supply chain risk
Cx 1.1.1a The argument is derived from Howard paper
C 1.1.1.1 Supplier has governance policies and practices that support security
C 1.1.1.2 Supplier has effective processes in place that support development of secure products
C 1.1.1.3 Supplier has good practices for customer security problem response
C 1.1.1.4 The supplier has good policies and practices for evaluating the supply chain risks introduced by its suppliers
Cx 1.1.1.4a This is a recursive use of this case
Cx 1.1.1.1.1a These are the standard set of good practices for the physical development environment
C 1.1.1.1.1 Supplier has good policies and practices to ensure development site security
C 1.1.1.1.2 Supplier employees are educated as to security engineering practices
Ev 1.1.1.3.1 Description of process for responding to security problems
Ev 1.1.1.3.2 Description of the defined point of contact for failure/bug reports
C 1.1.1.1.2.1 All engineers are educated/trained
C 1.1.1.1.2.2 Training is updated sufficiently frequently
C 1.1.1.1.2.3 Appropriate experts teach the classes
Ev 1.1.1.1.2.1.1 Documentation for each engineer of training received and when trained/retrained
Ev 1.1.1.1.2.2.1 Revision dates for training materials
Ev 1.1.1.1.2.3.1 Lists of acceptable credentials for instructors
Ev 1.1.1.1.2.3.2 Names of instructors and their credentials

Figure 6: Assurance Case for Supplier Practices

Claim C1.1.1.1 addresses the governance practices that are called out in the BSIMM. The BSIMM is particularly concerned that the organization has good policies and practices in place to ensure development site security and that employees are well trained in application software security engineering practices. The kind of evidence that this argument depends on includes training materials and confirmation that employees have indeed been trained. We have not documented criteria for evaluating the quality of the training materials or the nature of a satisfactory confirmation that employees have been trained, in part because there is no general agreement about what such criteria should be and in part to simplify the presentation of the case. In the absence of explicit criteria, reviewers will have to use their own judgment as to whether the evidence is of sufficient quality to support a specific claim.

Next, we consider claims C1.1.1.3 and C1.1.1.4. The first of these claims requires evidence that the supplier has a process and a point of contact for dealing with customer-reported security issues. Claim C1.1.1.4 requires evidence that the supplier has good practices for evaluating the supply chain security risks introduced by any suppliers that *it* contracts with. This would involve a recursive invocation of this assurance case on that supplier.

The argument shown in Figure 7 supports claim C1.1.1.2 and is quite a bit more detailed than those for the claims just discussed. Claim C1.1.1.2 says that the supplier has effective processes in place that support the development of secure products. The argument showing this to be valid requires the supplier to

- have design practices that make the product more robust against security threats (claim C1.1.1.2.1)
- follow suitable security coding practices (claim C1.1.1.2.2)
- perform good testing and V and V (validation and verification) (claim C1.1.1.2.3)
- have a process for documenting security aspects of its products (claim C1.1.1.2.4)

Showing that the supplier has design practices that make the product more robust against security threats (C.1.1.1.2.1) requires that the supplier have documented design guidelines that show that appropriate security design principles are used. It further requires that threat modeling (Section 2.3) techniques be applied to the design.

Showing that the supplier follows suitable security coding practices (claim C1.1.1.2.2) involves detailing the compilers and other tools used to develop the software to ensure that they are of suitable quality. High-quality compilers have built-in checks to avoid common mistakes that may lead to insecure code. Disabling these checks greatly reduces the power of these compilers, so the assurance case requires that the appropriate checks be enabled. Suitable security coding practices also require the use of static analysis tools throughout development, and thus the case requires evidence of their use and efficacy. Other important indicators of suitable coding practices are evidence that dangerous application programming interfaces (APIs) are avoided and that the supplier employs encryption when protected information could be intercepted.

This portion of the case applies mainly to software developed in-house or commissioned through a supplier. For COTS/GOTS components as well as for free and open source software (F/OSS), one can attempt to obtain information about supplier development practices, but much of the necessary data will simply be unavailable. In such cases, the software acquirer can, at the very least, attempt its own V and V activities. In this instance, penetration testing (claim C1.1.1.2.3.1) and fuzz testing (claim C1.1.1.2.3.2) can be extremely valuable.

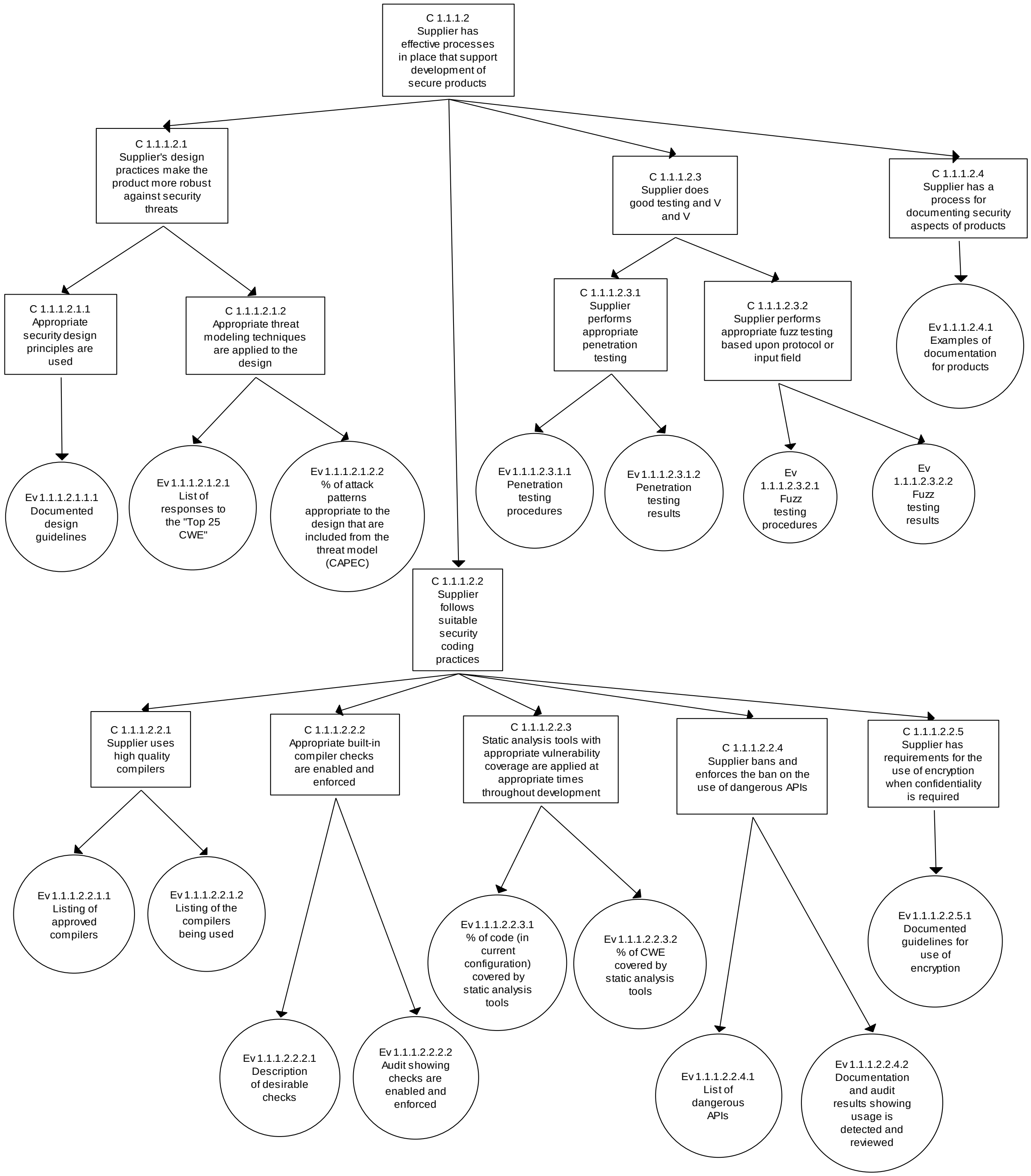
C 1.1.1.2 Supplier has effective processes in place that support development of secure products
C 1.1.1.2.1 Supplier's design practices make the product more robust against security threats
C 1.1.1.2.1.1 Appropriate security design principles are used
Ev 1.1.1.2.1.1.1 Documented design guidelines
C 1.1.1.2.1.2 Appropriate threat modeling techniques are applied to the design
Ev 1.1.1.2.1.2.1 List of responses to the "Top 25 CWE"
Ev 1.1.1.2.1.2.2 % of attack patterns appropriate to the design that are included from the threat model (CAPEC)
C 1.1.1.2.3 Supplier does good testing and V and V
C 1.1.1.2.3.1 Supplier performs appropriate penetration testing
Ev 1.1.1.2.3.1.1 Penetration testing procedures
Ev 1.1.1.2.3.1.2 Penetration testing results
C 1.1.1.2.3.2 Supplier performs appropriate fuzz testing based upon protocol or input field
Ev 1.1.1.2.3.2.1 Fuzz testing procedures
Ev 1.1.1.2.3.2.2 Fuzz testing results
C 1.1.1.2.4 Supplier has a process for documenting security aspects of products
Ev 1.1.1.2.4.1 Examples of documentation for products
C 1.1.1.2.2 Supplier follows suitable security coding practices
C 1.1.1.2.2.1 Supplier uses high quality compilers
Ev 1.1.1.2.2.1.1 Listing of approved compilers
Ev 1.1.1.2.2.1.2 Listing of the compilers being used
C 1.1.1.2.2.2 Appropriate built-in compiler checks are enabled and enforced
Ev 1.1.1.2.2.2.1 Description of desirable checks
Ev 1.1.1.2.2.2.2 Audit showing checks are enabled and enforced
C 1.1.1.2.2.3 Static analysis tools with appropriate vulnerability coverage are applied at appropriate times throughout development
Ev 1.1.1.2.2.3.1 % of code (in current configuration) covered by static analysis tools
Ev 1.1.1.2.2.3.2 % of CWE covered by static analysis tools
C 1.1.1.2.2.4 Supplier bans and enforces the ban on the use of dangerous APIs
Ev 1.1.1.2.2.4.1 List of dangerous APIs
Ev 1.1.1.2.2.4.2 Documentation and audit results showing usage is detected and reviewed
C 1.1.1.2.2.5 Supplier has requirements for the use of encryption when confidentiality is required
Ev 1.1.1.2.2.5.1 Documented guidelines for use of encryption

Figure 7: Supplier Practices Supporting Development of Secure Products

3.3 The Delivered/Updated Product Is Acceptably Secure

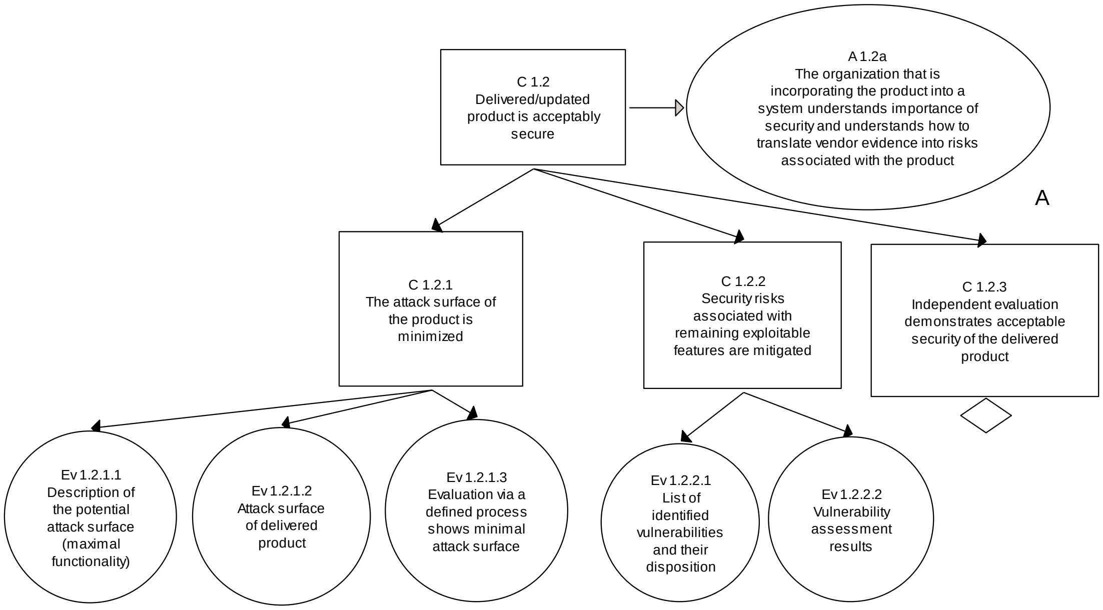
C 1.2
Delivered/updated product is acceptably secure
A 1.2a
The organization that is incorporating the product into a system understands importance of security and understands how to translate vendor evidence into risks associated with the product
A
C 1.2.1
The attack surface of the product is minimized
C 1.2.2
Security risks associated with remaining exploitable features are mitigated
C 1.2.3
Independent evaluation demonstrates acceptable security of the delivered product
Ev 1.2.1.1
Description of the potential attack surface (maximal functionality)
Ev 1.2.1.2
Attack surface of delivered product
Ev 1.2.1.3
Evaluation via a defined process shows minimal attack surface
Ev 1.2.2.1
List of identified vulnerabilities and their disposition
Ev 1.2.2.2
Vulnerability assessment results

Figure 8: The Delivered/Updated Product Is Acceptably Secure

Figure 8 expands claim C1.2. This claim states that the delivered/updated product is acceptably secure. We argue that the claim is valid if

- The attack surface of the product is minimized (claim C1.2.1).
- Security risks associated with the remaining exploitable features are mitigated (claim C1.2.2).
- An independent evaluation demonstrates acceptable security of the delivered product (claim C1.2.3, not further developed here).

Showing that claim C1.2.1 is valid requires providing evidence that the potential attack surfaces (as discussed in Section 2.2) have been determined and minimized. Showing that claim C1.2.2 is valid requires a detailed vulnerability assessment with a discussion of mitigations.

As in Section 3.2, the nature of the product being supplied plays a big factor in determining exactly what “acceptably secure” means. For a commissioned system, there should be complete visibility of all the required evidence. For COTS/GOTS components and F/OSS, the available evidence will probably be more opaque. However, even in this case, the supplier should be required to document what might affect the shape and size of the attack surface. For instance, the product may come preconfigured with open network ports and default passwords. They must be documented so that unused ports can be closed and default passwords can be changed to further minimize the attack surface.

3.4 Methods of Transmitting the Product

The methods used to deliver the product to its user guard against the introduction of malware in transit (claim C1.3 below, not further developed here; see Figure 9). This claim corresponds to the “product logistics” area mentioned in Section 1.2.

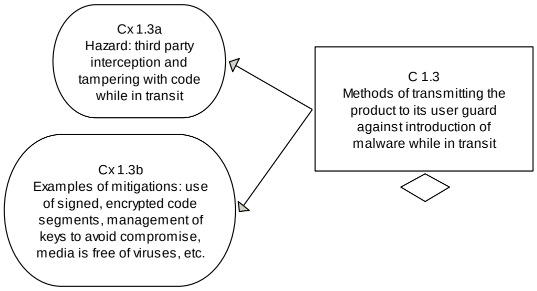
Cx 1.3a
Hazard: third party interception and tampering with code while in transit
C 1.3
Methods of transmitting the product to its user guard against introduction of malware while in transit
Cx 1.3b
Examples of mitigations: use of signed, encrypted code segments, management of keys to avoid compromise, media is free of viruses, etc.

Figure 9: Methods of Transmitting the Product to Its User Guard Against Introduction of Malware While in Transit

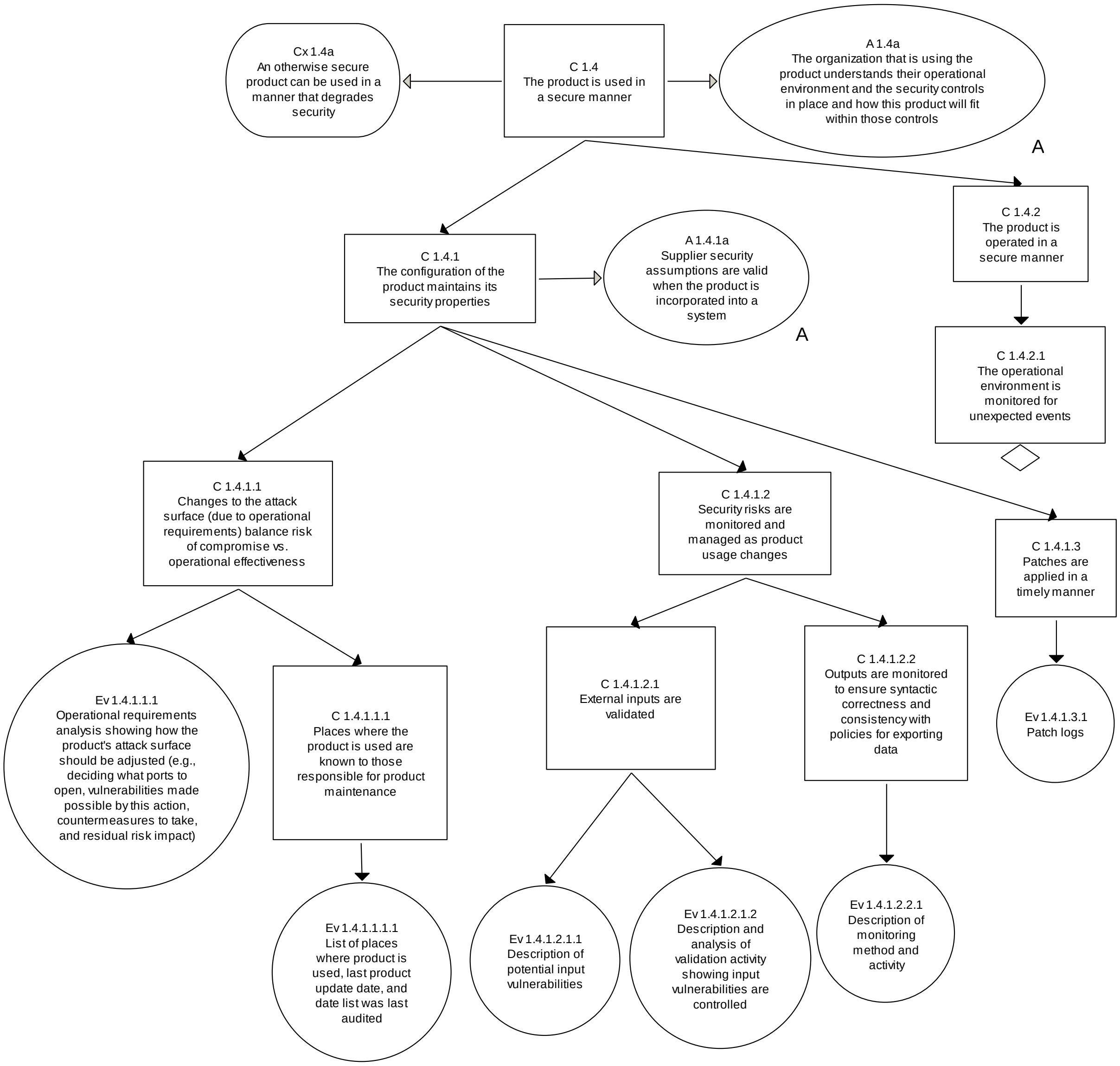
Cx 1.4a
An otherwise secure product can be used in a manner that degrades security
C 1.4
The product is used in a secure manner
A 1.4a
The organization that is using the product understands their operational environment and the security controls in place and how this product will fit within those controls
A
C 1.4.1
The configuration of the product maintains its security properties
A 1.4.1a
Supplier security assumptions are valid when the product is incorporated into a system
A
C 1.4.2
The product is operated in a secure manner
C 1.4.2.1
The operational environment is monitored for unexpected events
C 1.4.1.1
Changes to the attack surface (due to operational requirements) balance risk of compromise vs. operational effectiveness
C 1.4.1.2
Security risks are monitored and managed as product usage changes
C 1.4.1.3
Patches are applied in a timely manner
Ev 1.4.1.1.1
Operational requirements analysis showing how the product's attack surface should be adjusted (e.g., deciding what ports to open, vulnerabilities made possible by this action, countermeasures to take, and residual risk impact)
C 1.4.1.1.1
Places where the product is used are known to those responsible for product maintenance
C 1.4.1.2.1
External inputs are validated
C 1.4.1.2.2
Outputs are monitored to ensure syntactic correctness and consistency with policies for exporting data
Ev 1.4.1.3.1
Patch logs
Ev 1.4.1.1.1.1
List of places where product is used, last product update date, and date list was last audited
Ev 1.4.1.2.1.1
Description of potential input vulnerabilities
Ev 1.4.1.2.1.2
Description and analysis of validation activity showing input vulnerabilities are controlled
Ev 1.4.1.2.2.1
Description of monitoring method and activity

Figure 10: The Product Is Used in a Secure Manner

3.5 The Product Is Used in a Secure Manner

All the good design and supplier security practices in the world won't protect against misuse of the delivered product. If the product is shipped with a default, well-known password which is not routinely changed at installation, an easily exploited vulnerability exists. Claim C1.4, expanded in Figure 10, addresses this point. It argues that the product is used in a secure manner if it is

- configured to maintain its security properties (claim C1.4.1)
- operated in a secure manner (claim C1.4.2)

Claim C1.4.2 involves monitoring the environment for unexpected events and is not further developed here. Claim C1.4.1 is more complex and involves dealing with security issues that arise during operation of the product. Changes in operational requirements, in the way that the product is used, and to the product itself as patches are applied—all of these require analysis and monitoring to ensure ongoing security.

As in the previous two sections, COTS/GOTS components and F/OSS must be considered separately from commissioned systems. Usage changes, for example, can lead to the need to open additional network ports and may allow others to be closed leading to changes in the size and shape of the attack surface. Especially for COTS/GOTS software and F/OSS, this may lead to the need for additional validation and verification activities as well.

3.6 Putting It All Together

Figure 11 shows the entire supply chain security risk assurance case and how the pieces previously discussed all fit together.

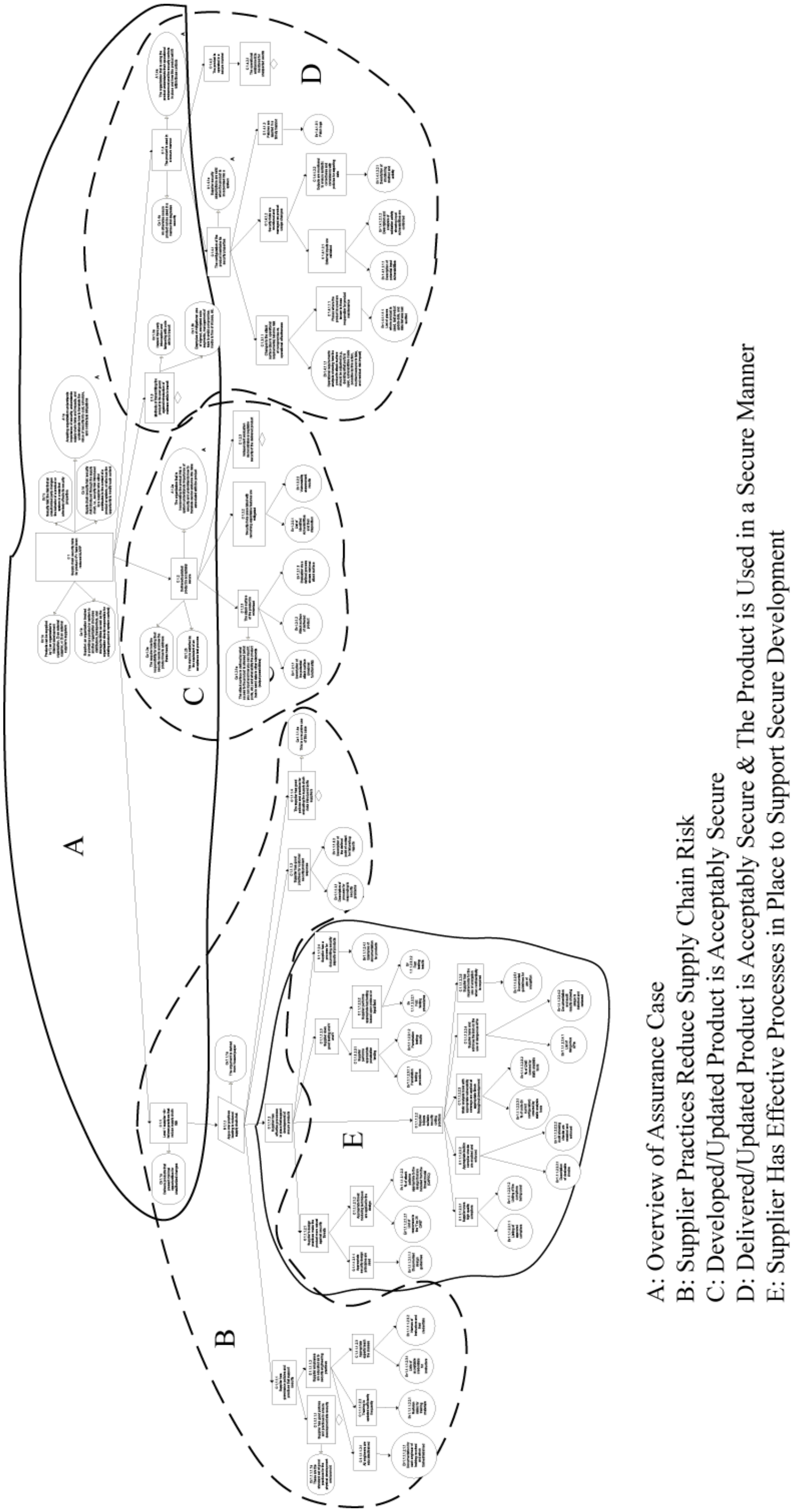
A
B
C
D
E
A: Overview of Assurance Case
B: Supplier Practices Reduce Supply Chain Risk
C: Developed/Updated Product is Acceptably Secure
D: Delivered/Updated Product is Acceptably Secure & The Product is Used in a Secure Manner
E: Supplier Has Effective Processes in Place to Support Secure Development

Figure 11: Putting the Assurance Case Together

3.7 Evaluating the Risk

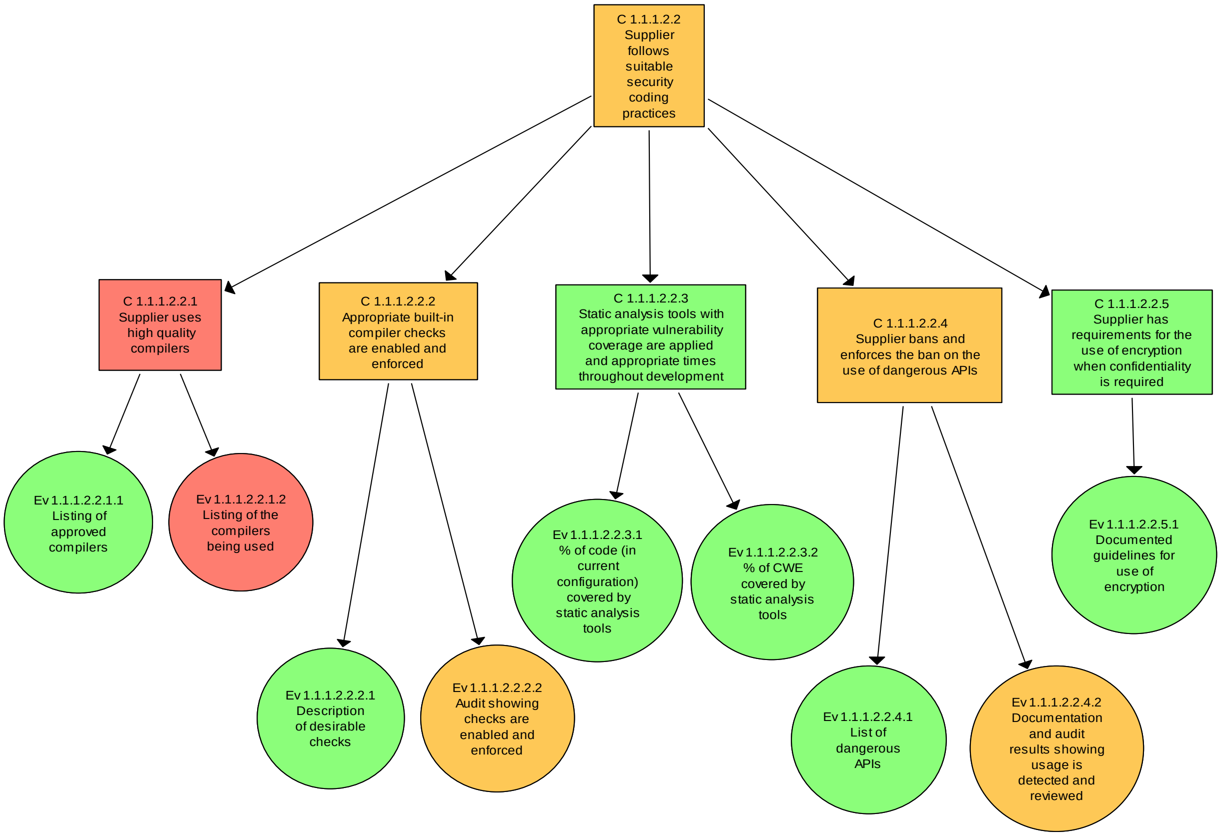
C 1.1.1.2.2 Supplier follows suitable security coding practices
C 1.1.1.2.2.1 Supplier uses high quality compilers
C 1.1.1.2.2.2 Appropriate built-in compiler checks are enabled and enforced
C 1.1.1.2.2.3 Static analysis tools with appropriate vulnerability coverage are applied and appropriate times throughout development
C 1.1.1.2.2.4 Supplier bans and enforces the ban on the use of dangerous APIs
C 1.1.1.2.2.5 Supplier has requirements for the use of encryption when confidentiality is required
Ev 1.1.1.2.2.1.1 Listing of approved compilers
Ev 1.1.1.2.2.1.2 Listing of the compilers being used
Ev 1.1.1.2.2.2.1 Description of desirable checks
Ev 1.1.1.2.2.2.2 Audit showing checks are enabled and enforced
Ev 1.1.1.2.2.3.1 % of code (in current configuration) covered by static analysis tools
Ev 1.1.1.2.2.3.2 % of CWE covered by static analysis tools
Ev 1.1.1.2.2.4.1 List of dangerous APIs
Ev 1.1.1.2.2.4.2 Documentation and audit results showing usage is detected and reviewed
Ev 1.1.1.2.2.5.1 Documented guidelines for use of encryption

Figure 12: Evaluating the Risk

Figure 12 (a fragment of the overall assurance case) illustrates one way of using the assurance case to evaluate the overall supply chain security risk. As evaluation proceeds, it may be impossible to locate some evidence, other evidence may be judged as imperfect, and still other evidence may be judged as perfect. Those evaluating the risk can color code the evidence according to their judgment. Nonexistent or poor evidence could be colored red. Solid evidence could be colored green. Anything falling between these two extremes could be colored yellow.

Coloring the claims supported by the evidence requires additional judgment on the part of the evaluator. Obviously if all the supporting evidence (or subclaims) have been colored green, it is reasonable to color the claim green, and if all the supporting evidence (or subclaims) have been colored red, the claim should be colored red. In the more usual case, the evidence and/or the subclaims will not be uniformly red or green. In such a case, the evaluator will have to decide the relative importance of the subnodes and determine an appropriate color for the blend—often yellow or red, but seldom green. Eventually the top node—“Level 1 supplier <S> follows practices that reduce supply chain security risk”—is colored with the overall evaluation of supply chain risk for the supplier.

4 Supply Chain Security Risk Review of a Current Program

Our team performed a preliminary review of the supply chain security risk for a selected DoD program using the assurance case reference model. The program selected for review is in the development phase of the acquisition life cycle. Our review was limited, since program funding was not available for the contractors to meet with us and most of the documents were not considered releasable to external parties. We were able to review the Software Development Plan (SDP) and conduct a group interview with key government members of the program office. **Note:** All quotes in this section are taken from the SDP.

4.1 Program Supply Chain

The supply chain structure for the program selected for review consists of a prime contractor that is focused on system and network management. Application and information assurance[16] software is provided by a major subcontractor to the prime. Three additional development companies are addressing specialized software needs and are subcontractors to the major software subcontractor. "The … Program deployment software consists of commercial off-the-shelf software, Government off-the-shelf software, Non-developmental software from Independent Development (ID)/Independent Research and Development (IRAD)s, and Free and Open source Software."

The COTS products were selected by the contractors as part of their initial response to the RFP. The contract is cost-plus and based on functionality to address the ORD. The architecture (service-oriented) was also part of the contractor bid. A great deal of the acquisition is outside of visibility of the DoD and the prime contractor. Exposure to the functionality is provided through periodic formal reviews specified in the SDP:

- System Requirements Review (SRR)
- System Design Review (SDR)
- Software Specifications Review (SSR)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)
- Test Readiness Review (TRR)

4.2 Evaluation of the Program's Supply Chain Security Risk

The supply chain assurance case is in four parts: (1) supplier capability, (2) product security, (3) product logistics, and (4) operational product control. In the remainder of this section, we present the program information provided for each of these parts.

[16] Information assurance software is software controlling access to data (e.g., software controlling the movement of data from higher to lower classification levels or software that mediates access via Common Access Card (CAC) readers).

4.2.1 Supplier Capability

Because the project is already in development, the contractor selection was complete, and we could not view information related to how that selection was made to identify considerations for supply chain security risk. However, we asked about supplier capabilities for the production of software with minimal security defects. The answer we were given described how the supplier was handling classified information and had no bearing on its ability to produce secure code.

Personnel are required to complete background checks and clearances to work in a top-secret environment. Developers are exposed to security awareness material, and an annual security briefing is conducted by security staff.

Security solutions are identified for system access and authentication but do not extend into the application software. The standard DoD solutions using public key infrastructure (PKI), certificates, role-based access controls, and intrusion-detection systems are included in the software architecture requirements. These can be supported either successfully or inappropriately by the software, depending on the skill of the developers. Security requirements mandate the use of application code signing, which at least provides a level of accountability if defects are appropriately tracked back to the source.

Based on information from the SDP, the skill focus is on a software developer’s ability to create new software. The following skills, which cover a typical list of programming capabilities, are required at various levels of experience, but it is not clear that knowledge of secure use of these tools beyond password control is expected: XML, C, C++, Java, CORBA, UNIX, ClearCase, Windows, Linux and Solaris, network administration, TCP/IP, X/Motif, DII COE, Simple Network Management Protocol (SNMP), Agent Technology, 3D(LDAP)v3 interfaces, OOA/OOD, UML, and COTS Integration. Without proper training, programmers using these tools can create code that allows all the common software attacks identified by the Common Weakness Enumeration. However, there are no widely accepted standards specifying what constitutes “proper” training, so training on appropriate coding techniques is often not considered to be a required part of a supplier’s capabilities. Although program coding standards are specified in the SDP to require use of Java, C, and C++, they do not include any consideration of secure coding standards for these languages.[17]

In addition, the subcontractors are building code using code-generation tools (e.g., Spring Framework) that will generate insecure code unless programmers have been trained to avoid these problems. The use of code-generation technology can increase supply chain security risk if security weaknesses are not properly addressed.

4.2.2 Product Security

There were no indications that software or supply chain security was considered beyond specific system requirements (PKI, certificates, role-based access controls, and intrusion-detection systems); for example, there were no requirements for using secure software development life-cycle

[17] For more information, go to http://www.cert.org/secure-coding/.

practices. Interviews indicated that the lack of emphasis on secure software development practices in current security regulations means such practices were not considered contractual requirements.

Interviews also indicate that a great deal of quality-of-service monitoring (for a SOA environment) is built into the infrastructure, and this monitoring can support security. Although there is extensive monitoring to ensure that software is well behaved, there was no evidence that the monitoring was designed to address security vulnerabilities identified by an attack surface analysis or by threat modeling.

Development is to apply "software engineering principles such as

1. isolation of interfaces to minimize data visibility and maximize information hiding
2. isolation of performance-sensitive software
3. encapsulation of COTS components
4. encapsulation of hardware and change points"

However, COTS tools used in development to generate code and COTS execution software that makes up the SOA infrastructure cannot be readily isolated. Many of these tools have security vulnerabilities. There are no formal acceptance criteria in place for COTS components. Control of patches and COTS product refresh is the responsibility of the contractor until the system is fielded, but there is no indication that a continuing review of potential security or supply chain concerns is required. For example, the impact of a vulnerability on an isolated service will be quite different from one that is heavily used. As a result, if usage changes or the service becomes more tightly integrated with the rest of the system, a new attack surface analysis and threat model should be done. There will be still greater concern for vulnerabilities that are within the SOA infrastructure on which all services rely. Planning for control of these varying levels of criticality was not evident.

"Testing shall be designed to not only show that all requirements are reflected in the software's behavior but also to try to demonstrate that the software's behavior is completely reflected in the design and requirements (no undocumented behavior)." The use of the term *try* does not provide a strong indicator of the level of success expected. That, coupled with limitations in encapsulation, indicates the potential for a very broad attack surface.

Because the prime contractor and subcontractors compete directly on other government contracts, exposure of identified problems through the supply chain has been problematic. Each supplier will inform the government but will not share that information with potential competitors. Though a formal reporting mechanism for problems has been formulated in the SDP, it is unclear how well this is being used based on the relationships of the contract participants.

The reviews are conducted by Quality Control, an independent auditing capability with a "separate reporting structure outside of the … program and engineering management structure." Theoretically this should provide the best level of information. However, interviews indicate that Quality Control personnel do not have the knowledge to cover everything, and additional outside support is being sought, although cost is a limiting factor.

4.2.3 Product Logistics

An informal control process is in place for all software elements during development, and all vendors share a central repository and other development tools. A formal control process under the control of a build coordinator is defined for the baselines of all products that move into the System Testing Phase. No information was provided about ensuring the integrity of transmitted code.

For limited user testing (LUT), the prime contractor hired a firm to formally control distribution of the software products from its environment to the specified user platforms and back. No information was available about the security practices used in distributing this software.

4.2.4 Operational Product Control

Control of security patches and COTS product refresh passes to post-deployment support beyond implementation. Patches are applied as soon as they are tested. Maintaining an effective SOA environment with minimum security risk that is highly dependent on COTS products customized for the SOA environment will present supply chain security risks beyond the normal vulnerability management concerns. No information was provided to show that these problems are being considered.

License management is carefully controlled, and control is transferred to the government at operational implementation. These licenses provide the government with access to support from the vendor, which includes security patches. Patch management processes were not yet defined.

Up to seven kinds of logging are available, but performance considerations must be factored into the use of each kind. Decisions of what is sufficient are still to be established. In addition, use of the logs will require some level of responsive monitoring that will be difficult, if not impossible, to do manually. The extent to which logging will support security reviews was unclear.

The fielded system is budgeted for five-year replacement of infrastructure components instead of the usual 20 years, which provides some protection against security deterioration. But even if patched, old versions of hardware and software frequently carry defects that are never fixed.

4.3 Review Conclusions

As a federally funded research and development center, the Carnegie Mellon® Software Engineering Institute has reviewed many DoD programs. Our review of this program showed that it was typical in its approach to software security and software supply chain security risk management. The DoD has focused on infrastructure security protections such as password-based access controls to prevent intrusion and has not developed regulations to mitigate security defects in application software. As systems become highly interoperable, the protection level provided by infrastructure security mechanisms diminishes. Without effective management, the software supply chain further increases opportunity for the introduction of defects that increase the risk of system compromise.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

The lack of appropriate contractual requirements makes it difficult for a program office to determine whether key supply chain security risk management practices are being followed. For example, assurance that application programmers are trained in secure coding and design practices will not normally be provided unless this is a requirement of the contract, and neither will information showing whether a supplier is following practices such as those suggested by the build-security-in maturity model or the security development life-cycle model. Furthermore, acquisition practices are currently weak in assuring that deployment practices (such as managing a deployed system's configuration to reduce its attack surface) and operational practices (such as preparing revised attack surface analyses and threat models) are adequate to maintain security against supply chain security risks.

5 Summary

Software supply chain security risks are only beginning to be recognized and addressed in DoD acquisitions for several reasons:

- The DoD has focused on infrastructure defenses against intrusions (i.e., firewalls, authentication mechanisms, etc.) and has not fully recognized the security risks posed by application software built on top of these mechanisms.
- Practices for defending against software supply chain security risks are relatively new and evolving, and they affect every phase of the acquisition life cycle.

To help the DoD develop and support better practices for mitigating software supply chain security risks, we examined risks and mitigation actions associated with each phase of an acquisition's life cycle. In particular, we noted that suppliers must be properly trained in software design, coding, and testing practices that address security vulnerabilities arising in application software and that program offices must monitor the extent to which these practices are actually used on specific developments. In addition, supply chain risk extends into a system's operational phase—at the minimum, suppliers and operational personnel need to provide responses as new threats are discovered and as new attack patterns arise; operational personnel also need to manage patches and the deployed system's configuration to ensure that no insecurities are introduced, either in the patching process or by improperly modifying configuration parameters.

We provided an initial version of an assurance case reference model describing the evidence needed to support claims that supply chain security risks have been adequately addressed throughout the acquisition life cycle. The reference model reflects our focus on two key strategies for controlling security risk: (1) identifying and monitoring a system's attack surface and (2) developing and maintaining a threat model. An implementation of these strategies requires different actions at different phases of the acquisition life cycle, and these differences are reflected in the assurance case reference model. Further work is needed to extend the model (e.g., to address the adequacy of supply chain security requirements) and to validate its usefulness.

Despite its preliminary state, the reference model served to guide our analysis of a specific DoD program. However, since the program was only in its development phase, the full scope of the model could not be used. Nevertheless, the structure of the assurance case provided a reasonable structure for interviews with program subject matter experts. By focusing on the specific areas of supplier capability, product security, product logistics, and operational product control, a broad range of program practices was considered in relationship to supply chain security risk.

References

[Allen 2008]

Allen, J. H., Barnum, S., Ellison, R. J., McGraw, G., & Mead, N. R. *Software Security Engineering: A Guide for Project Managers*. Boston, MA: Addison-Wesley, 2008.

[Codenomicon 2009]

Codenomicon. *Codenomicon DEFENSICS for XML Finds Multiple Critical Security Issues in XML Libraries*. http://www.codenomicon.com/news/press-releases/2009-08-05.shtml (August 5, 2009).

[Haimes 2008]

Haimes, Y.Y. *Risk Modeling, Assessment, and Management, 3rd Edition*. John Wiley & Sons, 2008.

[Howard 2003a]

Howard, Michael. "*Fending Off Future Attacks by Reducing Attack Surface.*" http://msdn.microsoft.com/en-us/library/ms972812.aspx (2003).

[Howard 2003b]

Howard, Michael, Pincus, Jon, & Wing, Jeannette. *Measuring Relative Attack Surfaces.* http://www.cs.cmu.edu/~wing/publications/Howard-Wing03.pdf (2003).

[Howard 2006]

Howard, Michael & Lipner, Steve. *The Security Development Lifecycle*. Microsoft Press, 2006.

[Kelly 1998]

Kelly, Timothy Patrick. "Arguing Safety—A Systematic Approach to Safety Case Management." PhD diss., University of York, Department of Computer Science, 1998.

[McGraw 2009]

McGraw, Gary, Chess, Brian, & Migues, Sammy. *The Building Security in Maturity Model.* http://www.bsi-mm.com/ (2009).

[MITRE 2009a]

The MITRE Corporation. *CWE-20: Improper Input Validation*. http://cwe.mitre.org/data/definitions/20.html (2009).

[MITRE 2009b]

The MITRE Corporation. *2009 CWE/SANS Top 25 Most Dangerous Programming Errors*. http://cwe.mitre.org/top25/index.html (2009).

[NIST 2002]
National Institute of Standards and Technology. *Risk Management Guide for Information Technology Systems.* Special Publication 800-30, July 2002,
http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[OWASP 2009]
Open Web Applications Security Project. *Software Assurance Maturity Model*.
http://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model (2009).

[SAPPWG 2008]
Software Assurance Processes and Practices Working Group. *Process Reference Model For Assurance Mapping To CMMI-DEV V1.2*. https://buildsecurityin.us-cert.gov/swa/downloads/PRM_for_Assurance_to_CMMI.pdf (June 23, 2008).

[Simpson 2008]
Simpson, Stacy, ed. *Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today.*
http://www.safecode.org/publications/SAFECode_Dev_Practices1008.pdf (2008).

[Simpson 2009]
Simpson, Stacy, ed. *The Software Supply Chain Integrity Framework: Defining Risks and Responsibilities for Securing Software in the Global Supply Chain.*
http://www.safecode.org/publications/SAFECode_Supply_Chain0709.pdf (2009).

[Steingruebl 2009]
Steingruebl, Andy & Peterson, Gunnar. “Software Assumptions Lead to Preventable Errors.” *IEEE Security & Privacy 7, 4* (July 2009): 84-87.

[Swiderski 2004]
Swiderski, Frank & Snyder, Window. *Threat Modeling*. Microsoft Press, 2004.

[Wikipedia 2009a]
Wikipedia. *Supply Chain.* http://en.wikipedia.org/wiki/Supply_chain (2009).

[Wikipedia 2009b]
Wikipedia. *Supply Chain Risk Management*.
http://en.wikipedia.org/wiki/Supply_Chain_Risk_Management (2009).

[Wikipedia 2009c]
Wikipedia. *Information Security*. http://en.wikipedia.org/wiki/Information_security (2009).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. **AGENCY USE ONLY** (Leave Blank)	2. **REPORT DATE** May 2010	3. **REPORT TYPE AND DATES COVERED** Final

4. **TITLE AND SUBTITLE**

Evaluating and Mitigating Software Supply Chain Security Risks

5. **FUNDING NUMBERS**

FA8721-05-C-0003

6. **AUTHOR(S)**

Robert J. Ellison, John B. Goodenough, Charles B. Weinstock, Carol Woody

7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. **PERFORMING ORGANIZATION REPORT NUMBER**

CMU/SEI-2010-TN-016

9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. **SUPPLEMENTARY NOTES**

12A **DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified/Unlimited, DTIC, NTIS

12B **DISTRIBUTION CODE**

13. **ABSTRACT (MAXIMUM 200 WORDS)**

The Department of Defense (DoD) is concerned that security vulnerabilities could be inserted into software that has been developed outside of the DoD's supervision or control. This report presents an initial analysis of how to evaluate and mitigate the risk that such unauthorized insertions have been made. The analysis is structured in terms of actions that should be taken in each phase of the DoD acquisition life cycle.

14. **SUBJECT TERMS**

Software supply chain security risk, assurance case, software assurance

15. **NUMBER OF PAGES**

49

16. **PRICE CODE**

17. **SECURITY CLASSIFICATION OF REPORT** Unclassified	18. **SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	19. **SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	20. **LIMITATION OF ABSTRACT** UL