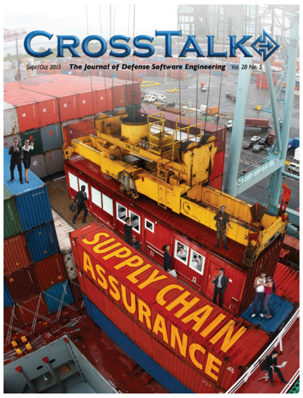
CrossTalk

Sept/Oct 2015 **The Journal of Defense Software Engineering** Vol. 28 No. 5

TABLE OF CONTENTS

Departments


Cover Design by Kent Bingham

Supply Chain Assurance

CrossTalk

NAVAIR	Jeff Schwalb
DHS	Joe Jarzombek
309 SMXG	Karl Rogers

Publisher	Justin T. Hill
Article Coordinator	Heather Giacalone
Managing Director	David Erickson
Technical Program Lead	Thayne M. Hill
Managing Editor	Brandon Ellis
Associate Editor	Colin Kelly
Art Director	Kevin Kiernan

Phone	801-777-9828
E-mail	Crosstalk.Articles@hill.af.mil
CrossTalk Online	www.crosstalkonline.org

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsor: Ogden-ALC 309 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit <www.crosstalkonline.org/subscribe> to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.crosstalkonline.org/submission-guidelines>. CrossTalk does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: CrossTalk is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services:
For questions or concerns about crosstalkonline.org web content or functionality contact the CrossTalk webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CrossTalk is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing <luminpublishing.com>. ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CrossTalk would like to thank **DHS** for sponsoring this issue.

This sponsor's note is taken from an interview with **Joe Jarzombek*** *after he spoke at CodenomiCON, when he shared his thoughts about supply chain assurance from a perspective of enterprise risk management and user safety and security.*

Imagine a cyber-reliant society with the Internet of Things (IoT) in which connected devices and products have been evaluated and certified from a perspective of consumer safety and protection. Safe and secure use of cyber assets places responsibilities on operators and users; yet realization of cyber assurance requires more focus on cyber safety and security in the supply chain. It requires network-connectable devices to be developed with cyber-physical security and safety in mind. IoT devices need to be patchable to be responsive to a changing threat environment. They need to be evaluated to determine they do not have malware, known vulnerabilities and software security weaknesses. This includes relevant certification activities that focus on mitigating exploitable weaknesses that could have otherwise been vectors of zero-day exploits if not mitigated prior to use. Independent third-party evaluation and certification is desired to assure relevant mitigations have been accomplished prior to use. IoT trust can be enabled with verification and validation activities focused on quality, safety, and security of devices in the context of the environments in which they would be used.

Fortunately for consumers, many white-hat researchers and test centers now provide third-party analysis of IoT products relative to cyber-physical security and safety. Underwriters Laboratories (UL) has launched its Cyber Assurance Program (addressing the needs stated above) and will be putting its mark on IoT devices and products; starting with healthcare systems, industrial control systems and network devices. These efforts provide better synergy between cyber assurance and cyber insurance since both provide a focus for mitigating residual risk.

The use of standardized cyber security terms is vital for information exchange needed in supply chain assurance. The UL CAP offers extensive coverage in evaluating IoT products for resilience to exploitation, and it adopts the use of several internationally recognized standards in the ITU-T CYBEX Series X for cybersecurity information exchange that covers data networks, open system communications and security. Standardized cyber security terms and enumerations enable interoperability, reduce ambiguity, and provide precision for managing efforts seeking to mitigate known vulnerabilities, exploitable weaknesses, and malware in cyber-enabled capabilities. The ITU-T CYBEX uses several DHS-sponsored standardized enumerations and languages vital to understanding the resilience of IoT products. The Common Weakness Enumeration (CWE™) https://cwe.mitre.org/ -- ITU-T CYBEX Recommendation ITU-T X.1524 http://www.itu.int/rec/T-REC-X.1524/en provides a formal list of known software-related weakness types – a specific type of mistake or condition that, if left unaddressed, could under the proper conditions contribute to a cyber-enabled capability being vulnerable to attack, allowing an adversary to make items function in unintended ways. A "weakness" represents a potential source vector for zero-day exploits or unreported vulnerabilities. Known weaknesses are CWEs – those characterized, discoverable, and potentially exploitable weaknesses with known mitigations. A "vulnerability" is a weakness with an associated exploit that can be directly used by an adversary to get a cyber-enabled capability to function in an unintended manner. Typically, this is the violation of a reasonable security policy for the cyber-enabled capability resulting in a negative technical impact. Although all vulnerabilities involve a weakness, not all weaknesses are vulnerabilities. The existence (even if only theoretical) of an exploit designed to take advantage of a weakness (or multiple weaknesses) and achieve a negative technical impact is what makes a weakness a vulnerability. Common Vulnerabilities and Exposures (CVE™) leverages common names and identifiers for publicly know information security vulnerabilities that have standardized use in ITU-T X.1520 CVE https://cve.mitre.org/. A vulnerability is a mistake in software that can be directly used by a hacker to gain access to a product, system or network. A configuration issue or a mistake in exposure is a vulnerability if it does not directly allow compromise but could be an important component of a successful attack and is a violation of a reasonable security policy. An information security exposure is a configuration issue of a mistake in logic that allows unauthorized access or exploitation. Known vulnerabilities are equated with publicly reported CVEs with patches in the National Vulnerability Database (NVD).

CVEs are easily discoverable through binary analysis; yet suppliers seem to routinely deliver new IoT products with old CVEs (some for which the patch has been available for more than four years). Two thirds of all CWEs are at the code level; detectable via static code analysis. Many tools are available to detect and aid in mitigating CVEs and CWEs. Why are users left on their own to find those CWEs and CVEs and mitigate or patch those products when developers could have easily mitigated the known vulnerabilities and weaknesses prior to delivery or as part of a product release update? Suppliers have no liability associated with products tainted with malware, known vulnerabilities and weaknesses. Why do some suppliers prohibit security researchers and users from evaluating products for these discoverable flaws that put users at risk?

Everyone can agree that IoT products need to have malware removed, and for supply chain assurance to be realized, suppliers, acquirers, and operators must also seek to mitigate known vulnerabilities and weaknesses prior to the products being put into use. The fact that new products are still being released with known vulnerabilities and weaknesses causes many to question the cyber hygiene of supply chain actors. It seems supply chain assurance can best be achieved with adoption of independent evaluation and certification of IoT products because realization of risks attributable to known vulnerabilities and weaknesses are primarily on the use side; not the supply side. Cyber insurance should also be interested in this cyber assurance practice since history has demonstrated that there seems little incentive for suppliers to change practices for mitigating these risks without independent evaluation and certification of IoT products.

*** Joe Jarzombek** *has been involved with CrossTalk for nearly two decades. As the Director for Software & Supply Chain Assurance (SSCA) in Cyber Security and Communications in the Department of Homeland Security (DHS) he leads public-private collaboration efforts for government interagency teams with industry, academia, and standards organizations focused on the assurance of ICT/software products and services. Through co-sponsorship of the SSCA Forum and Working Groups, he co-leads community efforts addressing cyber security needs, addressing software, supply chain external dependencies, and security automation initiatives to enable scalable information sharing among organizations and security researchers.*

Towards Supply Chain Information Integrity Preservation

Rasib Khan, University of Alabama at Birmingham
Md Munirul Haque, Purdue University
Ragib Hasan, University of Alabama at Birmingham

Abstract. In supply chains, it is important to preserve the integrity of a product as it is travels through the channel of distribution and is delivered to the final consumer. In absence of a secure mechanism, adversaries can manipulate supply chains to tamper with or introduce fake components. To mitigate this threat, we propose a novel architecture for preserving the integrity of the supply chain information system. We present the concept of asserted proofs and secure location governance to ensure tamper-proof supply chain information systems.

I. Introduction

Around the world, suppliers drive the overall product cycle. A product travels through a sequence of locations till it reaches the consumers, and is created by combining multiple intermediate products. There exist multiple business-to-business and business-to-consumer delivery channels in the production of each finished product. The sequence of destinations that a product travels through as it is manufactured and delivered to the final customer is referred as the product's supply chain.

The efficiency and security of the supply chain is a crucial concern for all industries. The transit of goods as it travels through the global supply chain system has critical effects on a nation's economy and security. Apart from disruptions in the supply chain, a nation can have highly unfavorable impacts with criminal and adversarial networks trying to exploit the system. We have seen that the supply chain in global economy has increased efficiency in recent times. However, products supplied from varied sources have introduced a greater risk in maintaining the integrity of the supply chain. Most works on supply chains include managing the supply chain, and the strategies used to optimize the process [1, 2]. References and models, such as the Supply Chain Operations Reference (SCOR) by Stephens [3], address the economic, financial, and managerial perspectives of the supply chain information management system. The process of validating and evaluating the supply chain performance can be a complex operation. There are numerous proposals on how to identify the necessary components to evaluate such supply chains [4]. However, little research has been done to allow secure generation of supply chain information in a secure and integrity protected manner.

In our work, we show how secure location provenance of a product helps to preserve the integrity of the supply chain [5]. Ensuring the integrity of the system implies an unforgeable preservation of location-specific records and information as a particular item travels through the supply chain. The supply chain provenance is the history of the product's locations over time. In generic term, it can be defined as the chain of custody (CoC), which refers to the "chronological documentation or paper trail, showing the seizure, custody, control, transfer, analysis, and disposition of physical or electronic evidence" [6].

Methods, involving continuous tracking and reporting of locations by third parties, violate the privacy and are not scalable for distributed environments. A more feasible and scalable approach is to require the product owner to obtain proofs of presence from each of the intermediate locations in the supply chain. To issue a proof, the authorities at a location first ensure the product's presence within a specific bounded region using secure localization techniques. A proof of presence can then be issued to the product owner, which can later be used to prove the product's location to a third party auditor. The supply of an item from the source to its final destination involves multiple intermediate locations and delivery authorities. Thus, a provenance chain is formed as the item travels through the supply chain. The provenance chain is delivered with the item at the final destination. The receiver of the supplied item can thus verify the obtained provenance chain, and validate the integrity of the item with respect to the intermediate locations, times, and chronological order of the visits for its supply chain.

II. Overview

Beginning with collection of raw materials to the finished product, the very nature of the freight life cycle provides ample opportunities to predators for replacing the authentic products with counterfeit items and tamper the supply chain records. With numerous hubs, assembly, and distribution points, it becomes a challenge to protect and actively monitor the supply chain. Here, we present the primary pitfalls of supply chain systems and the desired features of such a system.

A. Supply Chains – Risks and Pitfalls

The global supply chain system is dependent on an interconnected network of transportation, supplier, manufacturer, and information technology. As a result, the cross-operational entities allow significant risks across a broad geographic and industrial topology. A report published by The White House discusses the national strategies for global supply chain security and suggests active collaboration with the international community [9, 10]. The report discusses the strategies to promote the timely, efficient, and secure movement of goods, such that to preserve the supply chain from exploitation. It also suggests the requirement to improve verification and detection capabilities to identify contaminated, tampered, and prohibited items.

With an ever-increasing field of commercial activities, gray market distribution and monitoring counterfeit products have become a daunting task. The range of counterfeit products varies from relatively non-injurious products to serious health and safety related goods like medicine and insecticides. CNN reported in May 2012 that counterfeit electronic components from China have been incorporated into critical U.S. military systems [7]. This included operation helicopters and surveillance planes, which had put the troops at risk. The investigation for this case had actually been going on for a while before it was detected [8]. Over 80% of the active drug ingredients and 40% of medicines in the US are manufactured across 150 countries [15]. As a result, medicine and counterfeit drugs is another critical aspect of maintaining secure supply chain information systems.

Recently, Applied DNA Sciences Inc. proposed a DNA marking technology named botanical SigNature to ensure authenticity and guard against counterfeiting [11]. DNA taggants provide unique authentication scheme and forensic proof of provenance. But the biggest hindrance of this newest technology is the associated cost. Just the initial cost of procuring the DNA marking ink requires roughly 68,000 dollars [12]. Given the circumstances, we believe, a low cost, secure, and trustworthy digital solution for supply chain information integrity preservation is needed to effectively track and monitor the route of a product before it is delivered to the consumers. Additionally, access control and proper logging mechanisms should exist to keep track and monitor activities at the corresponding sites of the supply chain [13].

B. System Characteristics and Features

Distributed solutions are desired for many purposes, but they come along with certain requirements. Here, we provide the desired characteristics and features expected from an automated supply chain information integrity protection system:

- User Friendly: The user interface should be designed considering the background of the target people. Their familiarity with certain technology, education level, and physical condition should be considered during the design of the system in order to ensure maximum usability.
- Mobility: One key characteristic of the system is to maintain the mobility of the involved parties. Persons taking part in creating the proof of presence should be able to do that when they are in a specific bounded area rather than being present along with the product.
- Smoothness & Ease-of-Use: The system should be designed in such a way that users are comfortable with it. It should involve least possible human interaction ensuring that people will be interested in using the system for thousands of products each day.
- Extensibility: System design should ensure that some other useful services can be incorporated into the system later, i.e. the system should be extensible. Also it should be easy to incorporate suggestions from users during the later versions of the system.
- Efficiency: The system should cause little or no obstruction to the highly demanding activities of the supply chain. Timing requirement may very well define the success or failure of such systems.
- Cost Effectiveness: Supply chain involves literally thousands of products and millions of people. The overall cost for the whole system has to be marginal to expect a mass deployment of such a system around the globe.

In addition to the above-mentioned characteristics, such systems should have a certain feature set. Next, we present the desired feature set for a secure supply chain proof generation scheme:

- Secure Generation: The scheme should be secure and protected from adversarial threats. The generated proofs should therefore ensure absolute validity of the information presented within.
- Tamper Evident: In contrast to being tamper-proof, tamper-evident schemes ensure that any tampering with the proof should be detectable.
- Vulnerability: Any system might have a scenario where it fails to ensure security. However, any such system should be able to guarantee the minimum level of vulnerability in its operation and proof generation schemes.
- Information Assertion: In a secure scheme, a successfully generated proof should inherently imply that each of the entities has willingly asserted the information available within the proof.
- Chronological Order: The system should be able to ensure that the given set of proofs is chronologically ordered and the sequence is protected from tampering.
- Post-Validation: The proofs that are generated from the system should be able to be post-validated. That means, the proofs can be presented to an auditor, who can validate the integrity and validity of the proofs at a later time.
- Privacy Protected: The information in the supply chain system should be privacy protected, and must not be visible to any unauthorized personnel.
- Record Keeping Enabled: Once the proofs are generated, each of the entities must be provided with a copy of the proof for their own record keeping, which can be later retrieved if necessary.

III. System Model

In our work, we preserve the integrity of the channel of distribution for an item in order to mitigate the risks of counterfeit products [5]. The information for the supply chain is stored as a sequence of unforgeable and chronological location proofs, based on the path through which a particular product travels. The proposed scheme for preserving the supply chain information is based on certain entities and types of proofs in the system. In the following sub-sections, we present the elements of the system and the corresponding types of proofs required for modeling a secure supply chain information integrity preservation scheme.

A. System Elements

The following explains each of the elements in the system model, their purpose, and their functionalities.

- Product (P): A product is the item, which is being transported through the supply chain system. A product is identified with its bar code information, which stores the product identity and the product code. All proofs generated within the system include the product identifier.

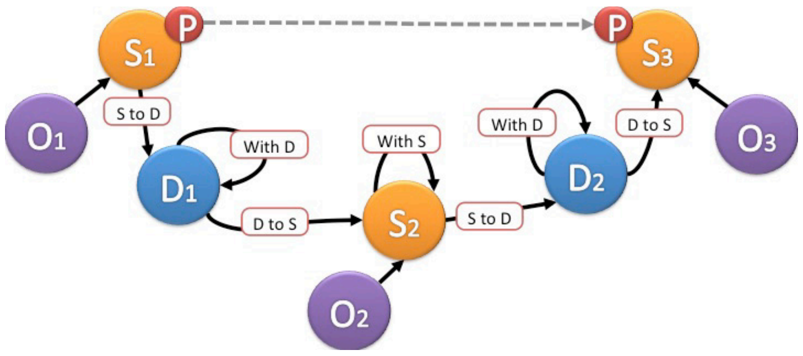


Figure 1: Points of Interest in the Supply Chain System. The figure illustrates a product transported from S1 to S3, and the corresponding points of interest to preserve the supply chain information.

• Site Authority (S): The site authority is the entity, which is responsible for any given site in the path of the supply chain system. A site authority can be a manufacturing authority, and actually manufactures a specific component for a larger product. Alternatively, a site authority can also be an intermediate authority, where the particular product has been located at least once, during the process of being delivered from the manufacturer to the final consumer.

• Operation Supervisor (O): Each site authority needs to have an operation supervisor, who is present on site during the delivery or dispatch of a product. Therefore, all manufacturing and intermediate authorities have an operation supervisor available for his services at the given site. The operation supervisor asserts valid deliveries and valid dispatches from a particular site in the supply chain.

• Delivering Authority (D): The delivering authority is the entity in the scheme, which is responsible for transporting a particular product between two destinations in the supply chain system. A delivering authority receives a particular product from a site authority, and delivers it to the next site authority according to the supply chain system.

• Auditing Consumer (C): The auditing consumer is the final recipient of a product. The auditing consumer is delivered the product along with a supply chain provenance. The supply chain provenance can then be validated to verify the claimed supply chain for the delivered product.

B. System Proofs

Given the activities within a supply chain, we are considering four different points of interest for our proposed scheme. As shown in Figure 1, a product P is being transported from site authority S1 to S3. In the supply chain system, the product actually is received from site authority S1 by a delivering authority D1. The product P is then transported to another location, and delivered to site authority S2. Subsequently, delivering authority D2 receives the product P, and delivers it to site authority S3. At each of the site authorities, there is also present an operational supervisor (O1, O2, and O3) to assert the delivery and dispatch operations.

Given the above context, we observed four points of interest where the integrity of the information should be preserved. According to Figure 1, the four points of interest for a product P are: (a) the product residing with a site authority, (b) the product being transferred from the site authority to a delivering authority, (c) the product residing with the delivering authority, and (d) the product being transferred from the delivering authority to another site authority. The four cases cover the possible scenarios while a product travels through the supply chain system.

Thus, given the four points of interest for preserving the integrity of the supply chain information, we have modeled the following proofs for the proposed scheme:

• Holding Proof: A holding proof is a logical evidence, which verifies the holding of a particular product at a particular site authority or a delivering authority.

• Transferal Proof: A transferal proof is a logical evidence, which verifies that a particular product has been handed over from a site authority to a delivering authority, or vice-versa.

• Supply Chain Provenance: A supply chain provenance is an alternating sequence of holding proofs and transferal proofs, which is chained together, such that the order of the sequence cannot be altered. The provenance chain can thus be utilized to prove the sequence of sites that a particular product has traveled.

C. System Capabilities

We assume that each site authority S has a server and WiFi network establishment. Additionally, the operation supervisor O and the delivering authority D carries mobile devices, which are capable of communicating with other devices and site authorities over WiFi networks. The devices have local storage for storing the supply chain proofs. It is assumed that the owner has full access to the storage and computation of the device, can run an application on the device, and can delete, modify, or insert any content in the data stored in the device. Additionally, it is assumed that the site authority, operation supervisor, and the delivering authority can access each other's public key.

According to the scheme, the site authority selects a supervisor from the list at random and sends a request to assert a proof for the given product P. Upon completion of a schematic communication among all the parties, each entity receives a proof, which has been mutually asserted by the other entities. Based on the context, the proof can either be a holding proof or a transferal proof. At a later time, the auditing consumer uses the individual proofs from the supply chain provenance and the yielded assertions in the proofs to determine the validity of the claimed locations in the supply chain system.

It should be noted that, we are trying to preserve the trace of chain of custody. Any question regarding the desired functionality of the product has not been evaluated here. Moreover, if the integrity of the product has been compromised before it has reached the first node of the supply chain, our model will not be able to detect that. Our focus is to integrate a digital provenance chain system which can be validated by the auditor at the receiving end and provide the information about the appropriate link of the supply chain records in case a product has been compromised. Additionally, each of the product and/or components is demarked with their corresponding barcodes, and is assumed to be unforgeable. That is, it is assumed that the item cannot be switched with the bar code remaining the same as before.

D. Threat Model

We consider different classes of adversaries, and also combinations of these adversaries in a collusion attack to exploit the integrity of the supply chain. In our threat model, we lay out the assets of the supply chain system and the capability of attackers. An adversarial entity in this context refers to any outsider, or an insider, who has an ill intention of modifying the information within the supply chain records.

The two main targets considered in our threat model are the place and time of the corresponding proofs within the supply chain records, both of which correspond to a particular product P. An adversary should not be able to create a proof for any site authority or delivering authority, where the product P has not ever been located. Also, even if the product P has been held by a specific authority, an adversary should not be able to create a proof for a different (local) time than the actual time of holding. Thus, a false proof of presence for a product P is one that asserts to the product P's presence at a location, which has not been visited by the product, or for a different time than the actual time of visit.

IV. Securing the Supply Chain

In our scheme for preserving the integrity of supply chain information, we address two specific types of proofs. In each of the cases, there are two individual scenarios, which must be addressed to secure the supply chain information. In the following, we discuss the four different proofs and constructing a tamper-proof provenance chain for the supply chain information system. A secure chain of the corresponding proofs, i.e., the provenance, will therefore guarantee tamper-resistance to the supply chain records. Moreover, we also show how the provenance chain is generated and verified by auditing authorities/customers.

In each of the cases, as shown in figure 2, we employ a three-way mutually authenticated interaction between the site authority SX, the delivering authority DX, and the operation supervisor OX. The three-way protocol allows a secure proof generation between the delivering authority DX and the site authority SX, which is asserted by the operation supervisor OX.

A. Holding Proof Generation

A holding proof refers to a proof of possession of a product at a particular time, and implies the responsible entity, which is accountable for the product at the given time. Thus, as a product travels through the supply chain, the product may reside with either a site authority or a delivering authority.

A. Holding Proof for Site Authority: In this occasion, the site authority SX is in possession of the product P and receives an asserted holding proof accordingly. The site authority SX initiates the process for generating an asserted holding proof. Subsequently, the delivering authority DX and the operation supervisor OX create the asserted holding proof, and send the proof to the site authority SX. After all the phases have completed, the site authority compares the two copies of the asserted proof: the one that was received directly from the operation supervisor OX, and the other one that was received via the delivering authority DX. If both the copies correspond to each other, the holding proof for product P at SX has been successfully generated. In case there is a failure in matching the two copies, the site authority SX issues an invalid assertion notification to the delivering authority DX and the operation supervisor OX, and discards the proof accordingly.

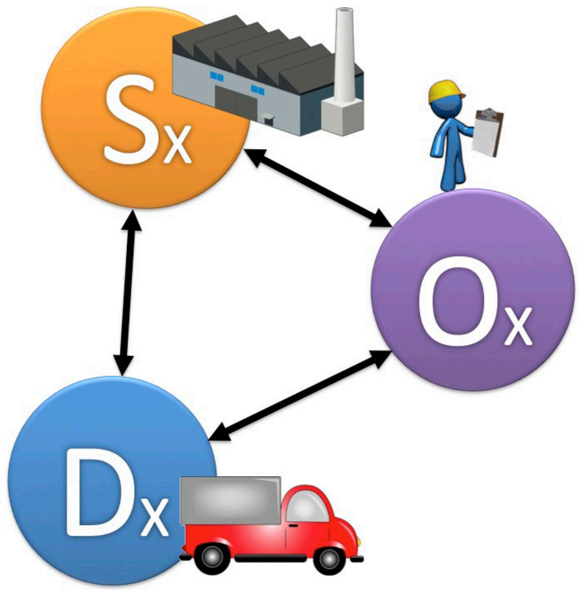


Figure 2: The Three-Way Interaction for Proof Generation. In each of the cases for generating location proofs, we employ a three-way mutually authenticated interaction.

B. Holding Proof for Delivering Authority: In the second case, the delivering authority DX is in possession of the product P, and receives a holding proof accordingly. The sequence of actions and messages are similar to the procedure described above. However, in this case, the delivering authority DX initiates the process for generating an asserted holding proof. Subsequently, the site authority SX and the operation supervisor OX creates the asserted holding proof, and sends the proof to the delivering authority DX. Similar to the method described previously, the delivering authority DX compares the two copies of the asserted holding proof. If successfully validated, DX stores the proof, or discards the proof otherwise.

B. Transferal Proof Generation

The transferal proof is a logical statement, which validates a successful transfer of authority and responsibility for a particular product P at a specific time. Implicitly, the transferal proof implies the release of liability of the product from a certain party. The transferal proof can be generated in two cases: the product P has been transferred from the site authority SX to the delivering authority DX, or, from the delivering authority DX to the site authority SX.

A. Transfer from Site Authority: In this case, the transferal proof is generated at the time when the product P is being handed over by the site authority to the delivering authority. The site authority sends a request for generating a transferal proof. The request includes the 'offload statement' from the site authority SX, which implies that the product P is being dispatched from the site after it was held in possession by the site authority SX for a specific duration of time. At the same time, the delivering authority receives the product P, and issues an 'onload statement' to the operation supervisor OX. The site authority SX then receives an asserted transferal proof with the 'onload statement' from the delivering authority DX, and the delivering authority DX receives an asserted 'offload statement' from the site authority SX. After successful completion of the above steps, the site authority SX and the delivering authority DX store the corresponding transferal proofs for future records. Each copy of the transferal proof bears an assertion from the operation supervisor OX. In case any of the validation procedures failed, the proof is discarded, and an invalid assertion notification is sent to the other entities in the system.

B. Transfer from Delivering Authority: The procedure for generating a transferal proof during the transfer from a delivering authority to a site authority is similar to the method described above. Instead of the site authority initiating the process, in this case, the delivering authority sends a request for generating a transferal proof. The request includes the 'offload statement' from the delivering authority DX, which implies that the product P is being offloaded from the delivering authority DX, after it was held in possession by the delivering authority DX for a specific duration of time. At the same time, the site authority receives the product P, and issues an 'onload statement' to the operation supervisor OX. Similar to the previous process, the delivering authority DX receives an asserted transferal proof with the 'onload statement' from the site authority SX, and the site authority SX receives an asserted 'offload statement' from the delivering authority DX. At any point in the protocol, any failed verification results in an invalid assertion notification.

C. Maintaining a Supply Chain Provenance

The individual holding proofs and transferal proofs are generated at the corresponding locations according to the supply chain of a product. At each of the phases where the individual proofs are being generated, the previous proof can be first checked to ensure earlier detection of supply chain anomaly. However, even after the proofs are created validly, the individual proofs are not organized in any sequence. Thus, the supply chain provenance serves the purpose of chaining the proofs in a manner such that the order and sequence of the proofs cannot be altered. A holding proof for the site authority must be succeeded by a corresponding transferal proof from the site authority to the delivering authority. The next proof in the chain is required to be another holding proof for the delivering authority. Figure 3 illustrates a provenance chain for the supply chain system. A supply chain provenance, which exhibits the given sequence, is a valid claim for the supply of the product P from site authority S1 to S2.

In the proposed protocol, we have used signed hash chaining to preserve the sequence of the proofs to create the provenance of the supply chain system. A hash chain is a concept of creating hash values from a sequence of linked values using standard hash functions, such as SHA-256/SHA-512. The provenance chain in this case can thus be created accordingly. This ensures that the order and integrity of the sequence is preserved. Subsequently, the corresponding signed hashes can be presented to the auditing consumer, along with the proofs. The auditing consumer can then securely verify each of the holding proofs, transferal proofs, and their order of sequence by recreating the hash chain.

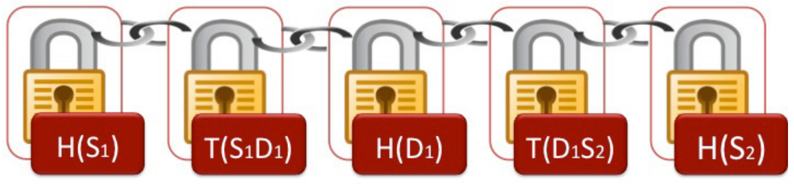


Figure 3: A Supply Chain Provenance. The figure illustrates a product transported from S1 to S2, and the corresponding proofs, which are being used to create a secure supply chain provenance.

V. Digital Content Supply Chain

Though our proposed model has mainly focused on protecting the chain of custody of tangible products, it also covers the domain of non-tangible products like software or intellectual properties considering their mode of transportation. Whenever any software or intellectual property transfers through the supply chain, they can be first converted into tangible goods like CD or paper documents, whose chain of custody can be maintained and proved by our system.

However, maintaining a secure supply chain system for digital content is also critical. Our proposed scheme can be adopted for digital content supply chain integrity preservation. In that case, the model would refer to the site authority as the content creator, the delivering authority as the content storage server, and the operational supervisor as the content approval authority. The three entities, therefore, would work in a similar fashion as described above. The digital content (e.g. software code) can be asserted by the content approval authority (e.g. software development team manager) for creating holding proofs for the content creator (e.g. software developer) and the storage server (e.g. code repository managed by the network admin). Additionally, the transferal proofs can also be created when a content creator downloads or uploads a digital content to or from the storage server. The provenance of the proofs, and hence the supply chain of the content, can be incrementally appended with the digital content (e.g. software product) as meta-data, and can be post-verified by an auditing authority and/or consumer using trusted certification authority signatures.

There are also known approaches to secure preservation of data provenance [14]. In such models, the generation, possession, and transfer of data are maintained using meta-file information. The meta-data are maintained in a secure provenance chain to ensure integrity preservation and tamper resistance. Therefore, we believe that similar data provenance techniques can be employed at the software code generation points to ensure a post-verifiable supply chain of program codes and components.

VI. Discussion and Future Work

The supply chain system is global system of diverse locations and items being transported over a network of suppliers, manufacturers, and delivery systems. The multitude of entities has introduced a greater risk in maintaining the integrity of the supply chain information system. In this work, we have described the motivation and desired properties of a secure supply chain recording process. Based on the given requirements, we have presented a model for the system elements and capabilities, system proofs, and the corresponding threat model. We utilized the model to design secure generation of holding proofs and transferal proofs, depending on the given context of actions within the supply chain. Furthermore, we also illustrated how the proofs can be used to create chronological hash chains. The given data item of provenance protected proofs can thus be effectively utilized to preserve the integrity of supply chain information and mitigate the risks of counterfeit components trickling into a system.

Currently we are testing the feasibility of our proposed model by deploying it on a small-scale supply chain. Our work so far has been able to validate the three-party assertion oriented proof generation. We have applied the designed model to generate proofs for locations and generated experimental results based on different threshold values. In our proof-of-concept prototype deployment, we were able to guarantee up to 99.99% reliability for the proof requesting entity, with approximately 6% false positives during verification. Our future work includes developing a standalone low-cost device, which can perform the three-way interactive proof generation and protect the integrity of the supply chain system.

Acknowledgment

This research was supported by the Department of Homeland Security Grant #FA8750-12-2-0254 and a 2012 Google Faculty Research Award.

ABOUT THE AUTHORS

Rasib Khan, M.Sc., is a graduate research assistant in SECRETLab and a Ph.D. candidate at the University of Alabama at Birmingham, USA. Khan was a NordSecMob European Union Erasmus Mundus Scholar, and received dual MS degrees in Security and Mobile Computing from Royal Institute of Technology (KTH), Sweden, and Aalto University, Finland in 2011.
Phone: 205-566-4546
Email: rasib@cis.uab.edu

Dr. Md Munirul Haque, Ph.D., is a research scientist at the Regenstrief Center for Healthcare Engineering, Purdue University. Previously, Haque was a post-doctoral fellow in SECRETLab at the University of Alabama at Birmingham. He was a member of Ubicomp lab, Marquette University, USA, from where he received his Ph.D. in 2013. Haque received his B.Sc degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 2003.
Phone: 414-614-9715
Email: mhaque@purdue.edu

Dr. Ragib Hasan, Ph.D., is a tenure-track Assistant Professor at the Department of Computer and Information Sciences at the University of Alabama at Birmingham. Prior to joining UAB, He received his Ph.D. and M.S. in Computer Science from the University of Illinois at Urbana Champaign in October, 2009, and December, 2005, respectively, and was an NSF/CRA Computing Innovation Fellow at the Department of Computer Science, Johns Hopkins University. Hasan has received multiple awards in his career, including the 2014 NSF CAREER Award and the 2013 Google RISE Award.
Phone: 205-934-8643
Email: ragib@cis.uab.edu

REFERENCES

1. J. T. Mentzer, W. DeWitt, J. S. Keebler, S. Min, N. W. Nix, C. D. Smith, and Z. G. Zacharia, "Defining supply chain management," Journal of Business logistics, vol. 22, no. 2, pp. 1–25, 2001.
2. Simchi-Levi, E. Simchi-Levi, and P. Kaminsky, "Designing and managing the supply chain: Concepts, strategies, and cases". McGraw- Hill United-States, 1999.
3. S. Stephens, "Supply chain operations reference model version 5.0: a new tool to improve supply chain efficiency and achieve best practice," Information Systems Frontiers, vol. 3, no. 4, pp. 471–476, 2001.
4. B. M. Beamon, "Measuring supply chain performance," International Journal of Operations & Production Management, vol. 19, no. 3, pp. 275–292, 1999.
5. R. Khan, M. M. Haque, and R. Hasan, "A secure location proof generation scheme for supply chain integrity preservation," in Proceedings of The 2013 IEEE International Conference on Technologies for Homeland Security, ser. HST '13, Waltham, MA, USA, 2013, pp. 446–450.
6. Chain of custody, Wikipedia. Online at: <http://en.wikipedia.org/wiki/Chain_of_custody>.
7. L.Shaughnessy, "Probe finds flood of fake military parts from China in U.S. equipment" Security Clearance, CNN Security News. Available online at <http://security.blogs.cnn.com/2012/05/22/probe-finds-flood-of-fake-military-parts-from-china-in-u-s-equipment/>. Tech. Report, May 2012.
8. P. Courson, "Report: Bogus U.S. military parts traced to China," CNN U.S. Online at <http://www.cnn.com/2011/11/07/us/u-s-military-bogus-parts/>, Tech. Report, Nov 2011.
9. The White House, "National strategy for global supply chain security". Online at <https://www.whitehouse.gov/sites/default/files/national_strategy_for_global_supply_chain_security.pdf>, Jan 2012.
10. The White House, "National strategy for global supply chain security implementation update," Online at <https://www.whitehouse.gov/sites/default/files/docs/national_strategy_for_global_supply_chain_security_implementation_update_public_version_final2-26-131.pdf>, Jan 2013.
11. Applied DNA Sciences, "Applied DNA sciences begins DNA marking microchips for a U.S. government agency - Protecting the electronics industry and government throughout the supply". Online at <http://www.marketwired.com/press-release/applied-dna-sciences-begins-dna-marking-microchips-for-a-us-government-agency-otcbb-apdn-1506279.htm>. Apr 2011.
12. Lee Mathiesen, "The microelectronics DNA marking moral dilemma". Online at <http://www.militaryaerospace.com/articles/2013/03/Mathiesen-DNA-viewpoint.html>, Mar 2013.
13. S.Borg, "Securing the supply chain for electronic equipment: A strategy and framework," Internet Security Alliance Publication, Nov 2008.
14. R. Hasan, R. Sion, and M. Winslett. "The case of the fake Picasso: Preventing history forgery with secure provenance". In Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST 09), pages 1–12. USENIX Association, 2009.
15. Howard Sklamberg, "Counterfeit Drugs: Fighting Illegal Supply Chains". Food and Drug Administration, Department of Health and Human Services. Online at <http://www.fda.gov/NewsEvents/Testimony/ucm387449.htm>. Feb 2014.

Software and Hardware Assurance

DoD Establishes Federation of Software and Hardware Assurance Providers

Tom Hurt, ODASD(SE)
Ray Shanahan, ODASD(SE)

Abstract. Keeping DoD hardware and software technology secure is more critical than ever. In response to a mandate from Congress, Deputy Secretary of Defense Robert O. Work chartered the Joint Federated Assurance Center (JFAC) [1] as a federation of U.S. Military Department and agency software assurance (SwA) and hardware assurance (HwA) organizations and capabilities. According to this charter, the JFAC is charged with supporting program offices throughout the life cycle with SwA and HwA expertise, capabilities, policies, guidance, and best practices. The JFAC is responsible for coordinating with DoD organizations and activities that are developing, maintaining, and offering software and hardware vulnerability detection, analysis, and remediation support. Other roles and responsibilities of the JFAC include:

- Conducting SwA and HwA analyses and assessments in support of defense acquisition, operations and sustainment activities;
- Advocating for the advancement of DoD interests in SwA and HwA research, development, and test and evaluation activities; and
- Building relationships with other communities of interest and practice in SwA and HwA such as other government organizations, academic environments, and private industry.

Introduction

The challenge of ensuring that DoD software and hardware will operate only as intended is formidable. DoD is more dependent than ever on technological solutions for mission requirements, and this has led to heightened awareness of the possibility that adversaries could target DoD supply chains, insert malicious functionality into software and hardware, or degrade critical systems with counterfeit parts. The globalization of the defense industrial base also has led to concerns about the competitiveness, cost-consciousness, and sources of many suppliers. Given the potential gaps in DoD SwA and HwA capabilities, as well as the cost and complexity associated with increasing the effectiveness of SwA and HwA throughout the life cycle of defense programs, DoD leaders seek to develop and promote enterprise solutions for evaluating and ensuring the cybersecurity of defense systems, components, and services, and for conducting remediation actions where necessary.

In the National Defense Authorization Act for Fiscal Year 2014, [2] Congress directed DoD to establish a joint federation of capabilities to support trusted defense systems and to ensure the security of software and hardware developed, acquired, maintained, and used by the Department. On February 9, 2015, Deputy Secretary of Defense Robert O. Work signed the charter for a new organization, the Joint Federated Assurance Center (JFAC), to coordinate this effort.

The JFAC builds on several earlier initiatives that also focused on strengthening the processes for assessing and implementing SwA, HwA, and related defense system trust and assurance activities. These activities include efforts to promote Trusted Systems and Networks (TSN), Supply Chain Risk Management (SCRM), and requirements for acquisition program managers to submit updated Program Protection Plans (PPP) at each milestone of the DoD acquisition life cycle. The JFAC will support these earlier initiatives and will enhance system security engineering (SSE) through DoD policy, guidance, studies, and supporting information products, as part of a comprehensive program protection process that promotes trust and assurance in defense system hardware and software.

JFAC Purpose and Objectives

As outlined in its charter, the JFAC will facilitate collaboration among the Military Departments and agencies that provide SwA and HwA services to ensure defense programs effectively plan, implement, and employ DoD SwA and HwA capabilities and investments throughout the acquisition life cycle.

The JFAC objectives include:

- Support program offices by identifying and facilitating access to DoD SwA and HwA expertise and capabilities. The JFAC will be a resource for program offices to access SwA and HwA policies, guidance, standards, acquisition practices, best practices, training, and testing support. In addition, the JFAC will provide access to assurance-related expertise and capabilities for DoD program offices, as well as facilitate coordination and support from the service providers.
- Identify and develop requirements for research and development (R&D) initiatives in support of the DoD R&D strategy to innovate vulnerability analysis, testing, and protection tools for SwA and HwA. Through a DoD SwA and HwA capability mapping process, the JFAC will identify potential gaps and needed capabilities.
- Enable efficient coordination and use of SwA and HwA design, analysis, and test capabilities. The JFAC will facilitate the exchange of information, techniques, and best practices for promoting assurance as part of the normal systems engineering and SSE processes.
- Serve as the DoD point of contact for interdepartmental and interagency efforts concerning SwA and HwA. The JFAC will engage with representatives of other federal departments and agencies as their access point to increase mutual awareness of tools, evidence-based practices, support environments, and an expanded talent pool.
- Develop and sustain a Department inventory of SwA and HwA resources, including tool licenses. The JFAC will explore and recommend ways to enhance access to enterprise licenses for selected automated software and hardware vulnerability analysis applications. The JFAC also will consider other potential ways to provide affordable and flexible access to automated, vetted tools for assessing and improving SwA and HwA throughout the Department.

Organization

Responsibility for management and oversight of the JFAC resides with a Joint Steering Committee led by the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)), in conjunction with representatives from the DoD Components. The ASD(R&E) is already charged with the development and oversight of defense policy and guidance for SSE, the Program Protection Plan Outline and Guidance [3], SwA, and HwA. The alignment of the JFAC with ASD(R&E) enables the JFAC to interact with SwA and HwA activities throughout the Department.

The JFAC Steering Committee is the governing body and provides senior-level management, oversight, and accountability for JFAC interests and concerns. Members of the Steering Committee currently include senior executive service level-representatives from the Office of the Under Secretary of Defense for Acquisition, Technology and Logistics; DoD Chief Information Officer; Departments of the Army, Navy and Air Force; Defense Information Systems Agency; National Security Agency; National Reconnaissance Office; Defense Microelectronics Activity; and Missile Defense Agency. Over time, the number of stakeholders may expand to take in additional defense organizations, issues and interests.

The JFAC currently includes three working groups and a coordination activity. The JFAC Action Officer Working Group is composed of senior staff from each organization within the JFAC Steering Committee, along with other members as approved by the JFAC Steering Committee. The JFAC SwA and HwA Technical Working Groups include subject matter experts representing the DoD service provider organizations and other technical expertise as needed. The JFAC Coordination Activity is composed of the JFAC Coordination Center (JFAC-CC) and representatives from the SwA and HwA service providers. The organizational structure is designed to facilitate open dialogue, coordination, and direct support for acquisition program managers from the federation of providers of SwA and HwA tools and services, throughout the life cycle of a program, with growing emphasis on sustainment.

Promoting Existing Assurance Capabilities

Each of the participating JFAC member organizations already manages an array of SwA and HwA capabilities and services as part of their normal program development, operations and support activities. Assembling these capabilities within a joint federated organization will bring about new opportunities to share information, promote best practices, prioritize and allocate scarce resources to address shared problems, and to inventory and license tools and resources. It will allow the organizations to standardize methods for identifying intelligence, research, development, and test and evaluation support for SwA and HwA interests and concerns.

Developing New Operational Concepts and Processes

The members of the JFAC working groups have already devoted considerable time and attention to developing a concept of operations for the federation that will encourage federation members to share information about their current SwA and HwA expertise, capabilities and capacities. For example, members can benefit from sharing information about relevant policies, standards, requirements, contract language, metrics, and procedures for acquiring, engineering, developing, testing, and evaluating trusted defense systems and services.

The members of the working groups are communicating with one another and their in-house service support organizations and other stakeholders about the roles and responsibilities of the federation in support of program offices, including applying the program protection planning process throughout the DoD acquisition life cycle. Improvements to the policy and guidance for SwA and HwA are being developed and will be applied at the next update of the Program Protection Plan Outline and Guidance, and evaluation criteria [4]. The JFAC member organizations are also developing individualized communication plans to outline process flows and methods for requesting services and support as part of the ongoing effort leading up to the declaration of Initial Operational Capability (IOC) for the JFAC.

At the declaration of IOC, which is expected to occur in the 4th quarter of calendar year 2015, the JFAC will be prepared to offer program management offices specific information about existing SwA and HwA service providers and capabilities, and guidance on how to plan and integrate these services and capabilities into their program management activities.

Assessing Capability Needs and Filling Gaps

Going forward from IOC to Full Operational Capability (FOC) over the next few years, the JFAC, in coordination with its member organizations providing SwA and HwA capabilities and services and related R&D efforts, will maintain a SwA and HwA capability map. The map will include a baseline of existing centers and capabilities of SwA and HwA services within the Department and elsewhere. The JFAC will identify and prioritize assurance capability gaps and will devise and recommend a strategy to validate and address such gaps. Potential gaps might include technical capabilities, resources and capacities, policy, assurance metrics, technical guidance, and program support tools or processes. If the necessary capabilities cannot be satisfied by existing centers and service providers, the JFAC working groups will make recommendations to the JFAC Steering Committee for consideration as the primary owners and users of assurance capabilities and services.

Fostering Cooperation

The JFAC is working with selected pilot programs nominated by their parent organizations to clarify the operational aspects of bringing together programs and assurance services without adding to the existing demands on program managers. The JFAC is committed to avoiding redundancy while helping programs identify and address SwA and HwA concerns that other quality control or testing activities might have overlooked. The pilots are looking at the Department's current SwA and HwA capabilities and interests and developing ideas regarding how the JFAC can best organize itself to support the needs of program managers, assurance service providers, and other stakeholders in an effective way.

The JFAC will align with and complement other SwA and HwA-related activities occurring in other parts of the Department, the U.S. government and industry, including the work of those involved in program protection, the DoD SwA Community of Practice, TSN, trusted suppliers, SCRM, systems engineering, SSE, and cybersecurity practice in the field, among others. The JFAC will work with and build upon these foundational activities to strengthen and promote trust and assurance in defense system hardware and software throughout the acquisition life cycle.

Promoting Trust and Assurance

In establishing its relationships with defense acquisition program managers and other stakeholders, the JFAC is committed to the following principles:

- A program's decision to participate in JFAC activities should be based on the need to bring about real and measurable improvement in the levels of trust and assurance for the program or system under development, throughout the life cycle of the program.
- The JFAC staff will seek to understand and adapt existing processes for providing SwA and HwA services to the programs to reduce the independent creation or reinvention of new processes and to control cost by blending-in best practices.
- The program's SwA and HwA needs should be specific, definable and measurable, and the assistance to be rendered by the JFAC should be within its established scope and set of capabilities.
- The JFAC will concentrate on identifying specific SwA and HwA areas of interest or concern that may not have been sufficiently addressed by program managers and other stakeholders.
- Although the JFAC's purpose is to share information, the JFAC does not intend to maintain sensitive information from programs and has established safeguards to protect sensitive program information against unauthorized release. The JFAC will share information only with those who have appropriate clearances, a need to know, and a responsibility for ensuring successful support and outcomes for the program and its stakeholders.

Supporting Needed Research and Technology Development

The JFAC is already making a positive difference in how the DoD advances SwA and HwA interests. As part of its initial assessment of existing SwA and HwA capabilities and gaps, the JFAC working groups identified several Department-wide needs for further research and technology development for SwA and HwA interests and concerns. The working groups recommended several technology development task proposals to the JFAC Steering Committee, which in turn approved and allocated funding. The results of these efforts will further the state-of-the-art for SwA and HwA facilities and organizations within DoD.

The JFAC Steering Committee also has allocated funding for analyses of SwA and HwA tools, techniques, and process. Tools such as the State-of-the-Art Resource (SOAR) for SW and HW, and the SOAR Matrix for existing SwA, provide guidance for selecting and using automated assurance tool sets across the DoD acquisition life cycle. The Steering Committee allocated funding to maintain and continue to improve the SOAR and other products of these DoD analyses for use by programs.

Engaging Other Communities of Interest and Practice

Moving forward, it is expected that the JFAC will continue to expand responsibilities to the full scope of the charter, including fostering closer cooperation with the academic community, private industry, and other federal government departments and agencies. Since R&D is a key component of JFAC operations, JFAC leaders will continue to identify, work with, and promote organizations dedicated to advancing innovative solutions for SwA and HwA inspection, analysis, detection, assessment, and remediation tools and practices.

Conclusion

DoD is establishing a joint federation of capabilities to support trusted defense systems and ensure the security of software and hardware developed, acquired, maintained, and used by the Department. As it continues to mature and develop, the JFAC will:

- Identify, operationalize, and institutionalize the Department's SwA and HwA capabilities in support of program management offices and other stakeholders.
- Evaluate the need for and impact of DoD investments in support of various SwA and HwA needs and interests.
- Collaborate across the DoD to influence R&D investments and bridge gaps in SwA and HwA capabilities.

Interested organizations are encouraged to contact the authors for more information about the JFAC. ◈

ABOUT THE AUTHORS

Tom Hurt is the Deputy Director for Software Engineering and Software Assurance, ODASD(SE). He is the DoD lead for planning, development, and establishment of the JFAC.
Phone: 571-372-6129
Email: thomas.d.hurt.civ@mail.mil

Ray Shanahan is the Deputy Director for Anti-Tamper and Hardware Assurance, ODASD(SE). He is the DoD HwA lead supporting planning, development, and establishment of the JFAC.
Phone: 571-372-6558
Email: raymond.c.shanahan.civ@mail.mil

REFERENCES

1. Deputy Secretary of Defense. Policy Memorandum (PM) 15-001–Joint Federated Assurance Center (JFAC) Charter. Washington, D.C.: Department of Defense, February 9, 2015. <http://www.acq.osd.mil/se/docs/JFAC-Charter-Signed-9Feb2015.pdf>
2. Section 937of Public Law 113-66, National Defense Authorization Act for Fiscal Year2014. 113th Congress, December 26, 2014.<https://www.congress.gov/bill/113th-congress/house-bill/3304/text#toc-H68884858A3434A2A92CC2F59FEC83EB6>
3. Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)). Program Protection Plan Outline and Guidance. Version 1.0. Washington, D.C.: DASD(SE), July 2011. <http://www.acq.osd.mil/se/docs/PPP-Outline-and-Guidance-v1-July2011.pdf>
4. Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)). Program Protection Plan Evaluation Criteria. Version 1.1. Washington, D.C.: DASD(SE), February 2014. <http://www.acq.osd.mil/se/docs/PPP-Evaluation-Criteria.pdf>

Premature Allocation of Program Requirements to Suppliers

Bohdan W. Oppenheim[1,2], Loyola Marymount University

Abstract. The paper presents a discussion of the difficulties of formulating stable requirements early in complex engineering programs, and the severe consequences on program execution. The problems are caused by the need to seek political and funding support for the program. Formal classical Systems Engineering (SE) and Program Management (CSEPM) methodology is based on the assumption that the knowledge to anticipate all interfaces and create good requirements exists early in the program, and it is only a matter of working out the details to build extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and high speed rail. The paper argues that this works well only for well-understood systems but it breaks down when the knowledge of what needs to be done still needs to be discovered, which is the case with most complex systems. In programs that develop new complex systems, the reality leads to the following Faustian Bargain: "Either develop and anticipate all interactions and requirements early, and allocate them to suppliers when the knowledge is not yet available, then conduct massive, painful, and cost-and schedule-busting requirements changes throughout the program; or delay the subcontracting until the system design is mature, complete and stable, and only then allocate requirements to subcontractors, but then risk the program termination because of the lack of political support and funding." The paper argues that in order to radically change this major deficiency of classical Systems Engineering and Program Management a radical change of the program business model would be needed.

1. Introduction

During the last 60 or so years, the classical Systems Engineering (SE) and Program Management (CSEPM) approach has evolved significantly, driven by two powerful forces: the uncompromising need for reliable system-level (including all subsystems) performance, and the inefficient government weapon acquisition practices. The evolution yielded a number of critical unintended consequences in CSEPM, including a shift from "great engineering" to "bureaucracy of artifacts", relying on massive outsourcing and inefficient mission assurance that involves premature requirement development, allocation and massive and costly requirements instabilities. The programs practicing CSEPM tend to achieve high levels of program success (e.g., 80 successful space launches in the U.S. Air Force [12]), but achieve this at the expense of notoriously costly and long (years and even decades) development programs, not infrequently with reduced performance.

This study is focused on requirements because they play a critical role in modern program formulation, execution and value/benefit delivery to customer stakeholders. It can be said that modern programs are driven by requirements. Yet, experience from complex programs such as satellite, spacecraft, ship, nuclear power plant, high speed rail, city infrastructure, and many others demonstrates that formulation of good and stable requirements is a formidable task and is rarely successful. In 2011 the Government Accountability Office [5] published an astonishing statistic that on average, 82% of requirements in recent defense programs are changed over the program lifecycle. That means that in spite of the huge effort, only 18% of the requirements released at the program initiation remain stable, a rather devastating number. This statistic is one reason for the notorious frustrations with large weapons and infrastructure programs, including exceeded program cost and schedule, Nunn-McCurdy reviews[3], and even premature program termination. Clearly, imperfect requirements are not the sole source of program troubles. Oehmen [9] lists 10 following Major Challenges in Managing Programs and each of them is capable of robbing a program of technical and/or business success:

1. Reactive Program Execution
2. Lack of stability, clarity and completeness of requirements
3. Insufficient alignment and coordination of the extended enterprise
4. Value stream not optimized throughout the entire enterprise
5. Unclear roles, responsibilities and accountability
6. Insufficient team skills, unproductive behavior and culture
7. Insufficient Program Planning
8. Improper metrics, metric systems and Key Program Indicators
9. Lack of proactive management of program uncertainties and risks
10. Poor program acquisition and contracting practices

Table 1 lists critical performance characteristics of nine major recent US Government space programs. The table data is based entirely on numerous GAO reports studied by [13]. The table indicates unstable requirements as a major contributor to program imperfect performance in seven of the nine programs listed. Besides unstable requirements GAO identifies the following other major reasons for program problems: unstable program funding (which is usually the result of other problems in a given program), starting the program before technology is sufficiently mature[4], and excessive complexity and features (named "gold plating of programs"[5] by Ashton Carter, then - Secretary of Defense for Acquisition and Logistics).

It is always desirable to correlate individual causes to program measures of success. Regretfully, the data quoted in Table 1 represents too small a sample size to allow that. The author was informed by his high-level contacts that that defense programs lack meaningful metrics of this kind. For example, government

"It is not realistic that all interfaces in a complex system can be anticipated and defined early in the program. Since all interfaces need be defined in order to write a complete set of requirements, it follows that it is not realistic to develop good detailed requirements at the program beginning."

PROG-RAM	Contr. Agency	Req's stable?	Funding stable?	# of TRL <<6	Final Cost B$	Cost Growth %	Schedule Growth%	# of Nunn-McCurdy Reviews	Excessive complexity?
SBIRS	Air Force	Unstable	Unstable	3	18.8	300%	120% Terminated	4	Yes
GPS IIF	Air Force	Unstable	Unstable	0	2.6	257%	133%	1+	No
GPS III	Air Force	Stable	Stable	0	4.2	2%	40%	0	No
GPS OCX	Air Force	Unstable	Unstable	14	3.695	28%	50%	N/A	Yes
MUOS	Navy	Stable	Stable	1	7.3	6%	20%	0	No
JMS	Air Force	Unstable	N/A	N/A	N/A	N/A	50%	N/A	Yes
SBSS	Air Force	Unstable	Stable	5	0.922	178%	60%	0	No
AEHF	Air Force	Unstable	Stable	11	14.372	154%	150%	3	No
NPOESS	Air Force, NOAA, NASA	Unstable	Stable	13	13.162	122%	Terminated	2	Yes

Table 1. Performance of Selected Major U.S. Space Programs [13]

programs do not track requirements instability over a program lifecycle, a critical measure of program quality[6]. The present paper is limited to a discussion of program requirements and related unintended consequences of the CSEPM evolution.

2.0 Unintended Emerging Properties of Classical Systems Engineering and Program Management

2.1 Premature Requirements and Massive Outsourcing

Developmental programs suffer from significant pressures to develop and allocate system requirements prematurely. The pressures are caused by the following two widespread practices:

a. Distribution of system development and production among as many geographically distributed suppliers as possible, driven by political pressures to “spread the wealth” in order to secure broad political support and funding for the program.

b. Overwhelmingly popular corporate policy to “stick to the system integration and subcontract the rest”, with the vast majority of system parts built by a complex network of suppliers.

More specifically, prime contractors of modern weapons and aircraft perform system design, major structural design and systems integration, and subcontract subsystems and components to the established vendor base, e.g. Boeing 787, F-35. It is normal for supplier network that builds a complex system to include thousands of vendors in four tiers of suppliers. The heavily outsourced and geographically distributed programs make the program coordination and system integration challenging and increase the need for excellent CSEPM. In a structure such as this, contracts, requirements and specifications with suppliers perform a critical role. Requirements and specifications prescribe the technical performance and interfaces between multiple parties, and typically fixed-price contracts define the budgets and schedule for each supplier. This association of requirements with cost and schedule is a large activity of CSEPM. With such a distributed network of design and production, all linked by legal and financial contracts, the only way to effectively produce systems is to develop excellent top-level requirements, then flow down and allocate them into subsystem requirements, which then flow down and allocate these requirements into component requirements and “build-to” specifications. The critical issue is that politics and funding require that all this activity be performed early in the program, before detailed knowledge about the system has been developed. Thus, immature allocated requirements are contracted to the suppliers, and then the system developers frantically iterate the design and change the requirements - which is a source of major program instability. The critical assumption in this approach is that knowledge exists early in the program to anticipate all system interfaces and perform intensive development of requirements, requirement allocations to subsystems, and program planning, and then just execute the program in a single cycle of requirements-allocation-design-build-integrate-verify-and-validate in order to deliver extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and submarines, and it is only a matter of providing enough resources to work out the iterations and details to create the needed system. The reality is that such formal techniques are effective only if designing a commodity for which significant legacy knowledge is available, but they break down when the knowledge of what needs to be done to write good requirements is lacking in early program phases, or the requirements are poorly formulated[7]. In this rigid system of thousands of fixed-price contracts with different suppliers in all tiers, any change of top-level requirements must be flowed down into all relevant suppliers and the contracts re-negotiated, typically with large delays, cost growth, or compromised performance in system development. The requirement verification and validation at all levels are the main tools of mission assurance in CSEPM. Unstable requirements and contested verifications are the notorious cause of arguments and legal actions between buyers and suppliers in the supply networks.

When a complex program starts with a large number of top-level requirements (and the recent trend is for increasing numbers, counted in low thousands), and when the technical knowledge of what is needed evolves over time slower than massive contracts with suppliers, the program instabilities cause significant "brute-force" iterations, information churning and thrashing due to the program pressures to keep requirements and test plans consistent – which tends to drive large cost and duration of complex programs. Almost all large governmental weapons programs demonstrate this behavior. The 82% of typical requirements being unstable mentioned earlier manifests that the single-pass execution of the classical SE process is not practical.

Hart-Smith [6][8] presented two additional powerful and well-substantiated arguments against massive outsourcing. Firstly, he documented the fact that outsourcing tends to outsource profits from a prime contractor to its subcontractors because subcontractors operate under fixed-price contracts; therefore the prime contractor has to absorb the costs of any design and requirement changes. Secondly, massive outsourcing introduces massive technical problems in system integration. This paper became quite controversial during the Boeing 787 aircraft development, which took the scope of outsourcing to extreme levels and experienced severe schedule and cost consequences.

2.2 Omitted Interfaces

In order to develop good requirements, all interfaces within the system and with system externalities must be identified. NASA SE Handbook [8] states (selected quotes from page 82):

"The bulk of integration problems arise from unknown or uncontrolled aspects of interfaces. Therefore, system and subsystem interfaces are specified as early as possible in the development effort. Interface specifications address logical, physical, electrical, mechanical, human, and environmental parameters, as appropriate....Interface specifications are verified against interface requirements...In verifying the interfaces, the system engineer must ensure that the interfaces of each element of the system or subsystem are controlled and known to the developers."

With n elements in the system, there are n(n-1)/2 possible one-to-one interfaces. A typical space vehicle or craft has tens of thousands of elements. This alone makes the interface definition effort formidable, as each interface is needed to write good specifications. Practitioners of SE anticipating interfaces understand the trepidation question "have we included all of them?" - knowing that even one omitted interface may cause fatal failure.

Particularly challenging are the interfaces involving humans. Armstrong, [1] stated that "human beings are naturally wicked[9]; therefore interfaces with humans are inherently wicked." In addition, most of the interfaces traditionally analyzed in technical systems are of the first order, with higher-order effects poorly understood and ignored. Two dramatic examples come from the two Space Shuttle tragedies. In the Challenger case, engineers understood that the rubber O-rings in the solid motor boosters must not be used in cold weather. They ignored the second-order human interface between the O-rings and the Shuttle flight management. The managers did not appreciate the risk of cold weather and ordered the flight, which led to the catastrophe, [2]. In the Columbia case, the interface between the foam covering the cryogenic tank and the airflow, as well as the secondary effect of the foam hitting and damaging the orbiter wings were poorly understood and ignored. The subsequent investigation determined that "the foam did it, the culture allowed it", [3]. Both above interfaces involving "management" and "culture" qualify as "wicked human interfaces." These interfaces were missing because the system was too complex to provide good insight and good understanding early in the program. And the more complex the system, the less knowledge is available just when it is needed.

It is not realistic that all interfaces in a complex system can be anticipated and defined early in the program. Since all interfaces need be defined in order to write a complete set of requirements, it follows that it is not realistic to develop good detailed requirements at the program beginning.

2.3 Model Based Systems Engineering is not the Solution for the Interface and Requirements Instability

The recently introduced elegant Model Based Systems Engineering (MBSE) approach [7;4] strongly automate and facilitate interface and requirements management, dramatically reduce the time, cost, error rate and pain of the SE process, and offer a number of other significant benefits, but MBSE is a tool of CSEPM and suffers from the same fundamental problems as the CSEPM: it cannot assure that all interfaces have been properly included, particularly the "wicked" ones. MBSE can help in identifying possible interfaces by making the n-squared matrix easier to manage, but cannot fill in the insightful details in each matrix cell. That task is still left to the experience and intuition of engineers. The problem is that the experience and intuition work well only for well-understood systems. It is not realistic that all interfaces in a new complex system can be anticipated and defined early in the program.

2.4 Bureaucracy of Artifacts

Another unintended consequence of the massive outsourcing is the complexity of the effort needed to coordinate development and production among a large number of parties. CSEPM solves the problem by having different organizations create Interface

Alternative 1 (CSEPM): Anticipate, develop and formulate all system interfaces and top-level requirements, then allocate them into subsystems and components, and sign fixed price contracts with numerous suppliers, each defining detailed technical specifications, cost and schedule - and do it all early in the program when the knowledge to do so is not yet available, driven by political support for government funding. Then, as you mature the system design, change the interfaces and requirements as needed. Regretfully, in this massively outsourced enterprise this requires changes in requirements to numerous subcontractors, and a massive bureaucracy of coordinating the work. One small change at the system level may trickle down into hundreds or even thousands of subcontracts requiring reworking of contracts, requirements, costs and schedules.

Requirements are changed not only because the knowledge needed to formulate good requirements is not available when they were initially documented. Great engineering must allow frequent opportunities for creative system - and subsystem - level improvements and optimizations. However, because the disturbance and overall cost of this reworking of the contracts tends to be severe, it usually occurs in traditional programs only when the program hits a major issue. As a result, such program changes are avoided by program management as much as possible. A consequence of this fact is that large complex systems are rarely optimized, which, in extreme cases, may place the system at serious risk. Oppenheim [12] quotes specific examples.

Alternative 2 (largely theoretical): No subcontracts are signed until the design is totally matured, completely defined and stable, including all interfaces. At this time the final and stable requirements are allocated to all levels of subsystems. Only then the subcontracts are signed. With stable requirements, subcontractors can verify and deliver system elements which can be integrated into the system predictably, and program cost and schedule are minimized. Regretfully, in this approach, the political support in the earliest program phases from the broad supplier base is missing, when it is politically needed the most to assure government funding. Without the funding, the program is still born.

Figure 1 "Faustian Bargain" of CSEPM

Control Documents (ICD) to document the interfaces among the program and system elements and coordinate the development. The ICDs are often complex documents that require months of work by tens of individuals each. Often, because of this long time scales, an ICD turns out to be obsolete on arrival, because system changes have taken place while the ICD was being created. Some programs spend years and significant treasure on such information churning. Instead of spending program effort and time on technical system optimization, program employees create massive ICDs. For this reason, CSEPM is said to have deteriorated from the early emphasis on "great engineering" to unintended "bureaucracy of artifacts."

2.5 Faustian Bargain

To summarize, the critical problem present in practically all programs creating complex systems is that the knowledge about the system which is necessary to define the top level requirements and their allocations is not available until both the system design and program are quite mature, but this is in conflict with political support, funding priorities, and outsourcing trends of programs. This leads the CSEPM to a "Faustian Bargain" described in Figure 1, of two equally bad alternatives, which is the critical unintended emerging characteristic of the CSEPM evolution during the last 40 years.

3.0 The Remedy

Let us accept that in the present climate of large government contracts the political and funding pressures require the practice of "spreading the wealth" among massive number of geographically distributed suppliers, otherwise the program risks the lack of funding. This, however, should not mean that premature requirements and specifications must be allocated to the suppliers before the design is mature and stable enough, which is bound to cause massive requirement and program instability. Quite the opposite: the prime contractor should perform complete system development and design to the level of "built-to package", and only then allocate the requirements, interfaces and production specifications to the suppliers. The proposed remedy is to create at the program beginning preliminary but binding agreements with suppliers for future work, thus assuring political support and funding for a new program, but to hold off with passing detailed specification to them until the design, interfaces, and the allocated requirements and specifications are fully mature, optimized and stable. The agreements should include promissory notes defining a minimum level of effort to be defined at a later date. This way, the hugely destructive requirements instability will be avoided, and programs will be enabled to execute predictably, stably and at minimum cost and schedule. Further, completing the design with full freedom from the contracts with suppliers is conducive to system optimization.

The proposed remedy, among others, has been captured in the so-called Lean Enablers for Managing Engineering Programs (LEfMEP], [9]. The publication includes 326 best practices which promote value to the customer stakeholders and reduce waste. The practices have been developed by integrating Systems Engineering, Program Management and Lean. A detailed description of LEfMEP is beyond the scope of the present paper.

We should also mention a radical solution undertaken by the privately owned rocket and spacecraft maker SpaceX: to be totally vertically integrated and not reliant on suppliers, and totally co-located, [12]. As such, there is no need to allocate and verify any requirements to suppliers. This business model has demonstrated extraordinary gains in system quality, development time and cost, but it is not practical to apply it to large government programs which inherently involve the "spreading the wealth" policy.

The system design should be performed by co-located teams with towering competence in the domain. If outside expertise is needed for the design, it should be brought into the team, rather than subcontracted out. Hart-Smith presents ample evidence of the destructive impacts on program health and system integration if system design is subcontracted out in pieces. Under no circumstances should the early part of system design, when the need for coordination is the strongest, be subdivided and outsourced to numerous vendors. Doing so is equivalent to cutting one's brain into pieces, sending them out to remote vendors, and then expecting that the pieces will function as a working brain.

4.0 Summary and Conclusions

Formal classical SE and PM (CSEPM) methodology is based on the assumption that the knowledge to anticipate all interfaces and create good requirements exists at the program initiation, and it is a matter of working out the details to build extremely complex devices such as satellites, aircraft, refineries, nuclear power plants and high speed rail. This works well for well-understood systems but it breaks down when the knowledge of what needs to be done still needs to be discovered, which is the case with most complex systems. In real programs that develop complex systems, the reality is more reminiscent of the "Faustian Bargain: "Either develop and anticipate all requirements and interactions early, when the knowledge is not yet available, and then conduct massive, painful, and cost-and schedule-busting requirements changes throughout the program; or delay the subcontracting until the system design is mature, complete and stable, and only then allocate requirements to subcontractors, but then risk the program termination because of the lack of political support and funding." As described in the text, the average number of requirements changes in large programs is 82%, which indicates that the first path dominates in industry. This Faustian Bargain is the unintended emerging characteristic of the CSEPM evolution during the last 60 years, driven by the geographical distribution of programs among a vast number of suppliers, which, in turn, is motivated by the politics of "spreading the wealth" and assuring program funding.

The paper identified other myths of modern CSEPM, including the assumption that all system interfaces can be anticipated early; and the naïve belief that distribution of design and production over a massive network of suppliers can be effectively coordinated with massive Interface Control Document bureaucracy.

The unintended evolution of CSEPM continues to worsen as the complexity of modern systems increases at significant rates. The constant dynamic of need, innovation and change makes it increasingly improbable that detailed and stable requirements can be developed at a program's initiation. This observation applies not only to space and national security programs but to a vast array of other complex government and commercial technology and socio-technological programs, such as cyber security systems, finance, internet communication, energy, nuclear waste, education, global warming, transportation, medical systems, and many others. In many of these programs the rational approach is to delay subcontracting specifications until the system design is mature and optimized, and requirements are stable. In addition, the design should be handles by a co-located team for ease of coordination and optimization.

The high cost and schedule penalty of renegotiating subcontracts in order to accommodate changing requirements and system optimization are not the only arguments against massive subcontracting. As [6] discussed from the perspective of a prime contractor, two other reasons are that massive outsourcing of value creation to numerous suppliers tends to outsource profits from a prime contractor to its suppliers, and introduces massive problems in system integration by the prime. The Boeing 787 program is an excellent example of highly excessive subcontracting.

The remedy proposed in the paper is to create at the program beginning a preliminary but binding agreements with suppliers for future work, thus assuring political support and funding for a new program, but to hold off with the passing of detailed specification to them until the design, interfaces, and the allocated requirements and specifications are fully mature, optimized and stable. The remedy, among others, has been captured in the so-called Lean Enablers for Managing Engineering Programs (LEfMEP], [9]. This approach should vastly increase program efficiency and weapon affordability.

Acknowledgement

The original study on which this paper was based was performed under the Project Management Institute (PMI) contract 14108 and published in [12]. The author is grateful to Mr. Stephen Townsend of PMI for his permission to include long fragments from the study in the present paper. The author wishes to thank Mr. John Muratore of SpaceX for generous sharing of the information which was used in the PMI study and from which the present paper strongly benefitted.

ABOUT THE AUTHOR

Bohdan "Bo" W. Oppenheim is a Professor of Systems Engineering at Loyola Marymount University in Los Angeles, and founder and co-chair of INCOSE Lean Systems Engineering Working Group, the largest Group of INCOSE. He co-led the project developing Lean Enablers for Systems Engineering (LEfSE), and served as an expert in the joint INCOSE-PMI-MIT project developing Lean Enablers for Managing Engineering Programs, integrating Lean Systems Engineering with Lean Project Management. Dr. Oppenheim authored the book Lean for Systems Engineering with Lean Enablers for Systems Engineering (Wiley, 2011), co-authored The Guide to Lean Enablers for Systems Engineering (PMI, INCOSE, MIT-LAI, 2012), and co-authored Lean for Banks; Improving Quality, Productivity and Morale in Financial Offices. He worked for five years at Northrop, four years at Aerospace Corporation, and consulted numerous defense and aerospace companies in the U.S. and Europe. His engineering degrees include Ph.D. from Southampton, U.K.; Graduate Engineer's Degree from MIT; MS from Stevens Institute of Technology; and B.S. (equiv.) from Warsaw University of Technology in Aeronautics. His industrial experience spans space, offshore, mechanical engineering and software, including several major aerospace and commercial programs. His credits include five books, numerous journal publications, $2.5 million in externally funded grants, and a 40-year industrial experience. He served on the teams honored with two Shingo Awards (2012 and 2013), and INCOSE Best Product Award (2010), as well as INCOSE Service Award (2014) INCOSE Fellow, 2015. The author published two former articles in CrossTalk.

NOTES

1. Professor of Systems Engineering, Loyola Marymount University, Los Angeles, boppenheim@lmu.edu. First submitted December 7, 2014.
2. This paper is based on a larger study [12].
3. A Nunn-McCurdy review is a mandatory congressional review for viability of a program when it exceeds 25% growth from its original cost.
4. The effect of contracting a program before it is ready from the technology readiness point of view was published in [11]. Programs are not supposed to proceed to the so-called Milestone B (design) unless all technology items are at Level 6-7 or higher. Yet, for expediency reasons many programs are contracted with immature TRL. (Wikipedia page <http://en.wikipedia.org/wiki/Technology_readiness_level> defines the TRL scale).
5. "Gold plating" refers to the program or system features and options which are excessive and unnecessary.
6. At the beginning of this study, a high-level manager from one of the Federally Funded Research and Development Centers supporting the U.S. Air Force Space and Missile Command offered staff time and budget to extract statistics on stability of requirements over program life cycle in major government space programs. After months of trying, it turned out such metrics are not collected and cannot be obtained without major effort.
7. Examples of bad requirements include sloppy, unclear, or incorrectly formulated requirements; not contributing to program value/benefit; mutually conflicted; parochially-motivated, not applicable to the system (e.g., the well-known case of submarine-relevant requirements being inserted into a spacecraft program); "gold-plated"; and the requirements that mandate that a particular subsystem design be included in the system, thus making system optimization difficult. Ideally, a rigorous independent review of all requirements should be performed after they are collected and before the requirements are released for RFP or contract. Highly competent and independent reviewers should catch all instances of unneeded and faulty requirements, catch the missing interfaces, push back on "gold plated" requirements, and demand that all these deficiencies be corrected before the program proceeds to the RFP or contract. Unfortunately, this is hardly ever performed.
8. The paper is labeled "Boeing Proprietary", however, it was leaked and published on the Seattle Times webpage <http://seattletimes.nwsource.com/AB-Pub/2011/02/04/2014130646.pdf>, therefore now exists in the public domain.
9. In this context, "wicked" refers to "unpredictable, mischievous."

REFERENCES

1. Armstrong J. R., "System Integration: He Who Hesitates is Lost", INCOSE Int. Symp., Las Vegas, 2014
2. Challenger Commission, Report to the President by the Presidential Commission on the Space Shuttle Challenger Accident, June 6th, 1986, Washington, D.C.
3. Columbia Accident Investigation Board (CAIB) Final Report on Aug. 26, 2003.
4. Friedenthal S., Moore A., Steiner E., Practical Guide to SysML, Second Edition: The Systems Modeling Language, The MK/OMG Press, 2012
5. GAO, DEFENSE ACQUISITIONS, "Assessments of Selected Weapon Programs", GAO-11-233SP, 2011.
6. Hart-Smith L. J., "Out-Sourced Profits - The Cornerstone of Successful Subcontracting", Boeing Third Annual Technical Excellence (TATE) Symposium, St. Louis, MI, February 14-15, 2001
7. Long D., Scott Z., "A Primer for Model-Based Systems Engineering", VITECH, Apr 4, 2012
8. NASA Systems Engineering Handbook, NASA/SP-2007-6105 Rev 1
9. Oehmen, J. Editor, The Guide to Lean Enablers for Managing Engineering Programs, PMI-INCOSE-MIT LAI, 2012
10. Oppenheim, B. W., Lean for Systems Engineering with Lean Enablers for Systems Engineering, Wiley & Sons, 2011.
11. Oppenheim B. W., "Improving Affordability: Separating Research from Development and from Design in Complex Programs", Crosstalk Journal, 25th Anniversary Issue, V. 26, No. 4, July/August 2013.
12. Oppenheim B. W., "Program Requirements: Complexity, Myths, Radical Change, and Lean Enablers", Project Management Institute, Paper 14108, 2015.
13. Pabst R.G., "Analysis of Management Practices in U.S. Space Programs using GAO data, and Mapping to Lean Enablers", Systems Engineering Capstone Project, Loyola Marymount University, Los Angeles, 2014
14. Wikipedia, 2014a: <http://en.wikipedia.org/wiki/Technology_readiness_level>
15. Wikipedia, <http://en.wikipedia.org/wiki/Load_testing> last accessed June 21, 2014

From DIACAP to RMF

A Clear Path to a New Framework

Major Henry R. Salmans III, USMC, Retired
Andrew C. Tebbe, MCICOM, USMC
William J. Witbrod, Computing Technologies, Inc.

Abstract. Department of Defense Instruction (DoDI) 8510.01, dated March 12, 2014, announced the adoption of the Risk Management Framework (RMF) for Department of Defense (DoD) Information Technology. The National Institute of Standards and Technology (NIST) Special Publication 800-39 fully articulates the RMF process which is a key input into DoDI 8510.01.
This article highlights what the transition from Department of Defense Information Assurance Certification and Accreditation Process (DIACAP) to "the RMF" means to Marine Corps "Information Assurance" and the DoD community at large[1]

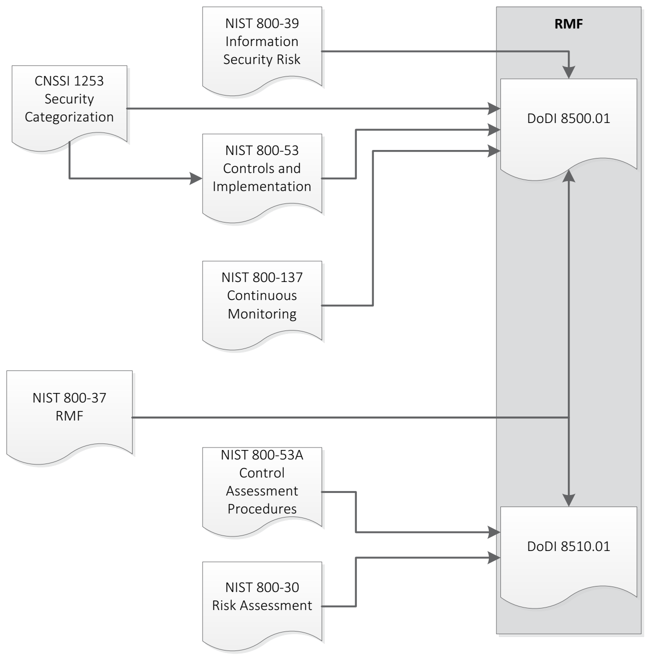

RMF
NIST 800-39 Information Security Risk
CNSSI 1253 Security Categorization
NIST 800-53 Controls and Implementation
NIST 800-137 Continuous Monitoring
NIST 800-37 RMF
NIST 800-53A Control Assessment Procedures
NIST 800-30 Risk Assessment
DoDI 8500.01
DoDI 8510.01

Figure 1: DoDI Publication Dependencies[4]

"Speed of action in cyberspace is critical to maintaining the advantage against adversaries and disruptions to service. Processes must be in place to facilitate this speed of action to allow for operational commander's mission needs while balancing security. The adoption of RMF hopefully further streamlines the critical accreditation of systems. One objective being to give commanders an ability to manage risk in cyberspace in a way that makes sense as in other warfighting domains."
-Colonel David W. McMorries (former Commanding Officer, Marine Corps Network Operations and Security Center)

RMF Transition

The DoD transition to the RMF is an evolution in the DoD Cybersecurity[2] program to address the changing risk to information systems. RMF is a Federal standard and DoD's adoption of it will enable greater interoperability, knowledge sharing, and reciprocity across the Federal government. Using a more robust system lifecycle approach for risk assessment, along with a more scrutinized continuous monitoring program, the Marine Corps can react more quickly and efficiently to changes within our Cyber environments. The RMF better aligns the DoD Cybersecurity language and practices with guidance provided by the National Institute of Standards and Technology (NIST) consistent for Federal information systems.

Although guidance from the Marine Corps regarding the transition to the RMF has not been released, the DoD has begun to update key instructions related to Cybersecurity under the RMF as presented in Table 1.

Figure 1 illustrates external publications used as the basis for the revised Cybersecurity Instructions.

The Knowledge Service Website[5], managed by the Department of the Navy for DIACAP, will be updated to reflect the transition to the RMF. The updated site serves as the authoritative source on guidance for implementing and executing the RMF according to the DoD Instructions and includes tools and templates for RMF execution and production of key artifacts.

Changes in Framework

Both DIACAP and RMF seek to identify and manage information system (IS) risks associated with system vulnerabilities and adversary threats. Vulnerabilities primarily consist of weak IS security procedures or internal controls. Threats exploit those vulnerabilities and include environmental disruptions, system or human errors, as well as purposeful attacks. The goal of both DIACAP and RMF is to mitigate vulnerabilities to an acceptable level of risk. Cybersecurity experts and practitioners transitioning from DIACAP will appreciate that the shared goal of risk management is equally true under RMF. Their knowledge and expertise, accrued under the previous framework, will be useful if not critical to the transition to this new paradigm.

Terminology

DoDI 8500.01 adopts the term "cybersecurity" throughout the DoD replacing "Information Assurance". The traditionally used Certification & Accreditation (C&A) process will be referred to

DoDI	Title	Reissue Date
8510.01	*Risk Management Framework (RMF) for DOD Information Technology*	03/12/2014
8500.01	*Cybersecurity*[3]	03/14/2014

Table 1: RMF DoD Instructions

as Assessment & Authorization (A&A) under RMF. Cybersecurity role titles have been changed, and in some cases responsibilities combined or divided among roles as presented in Table 2.

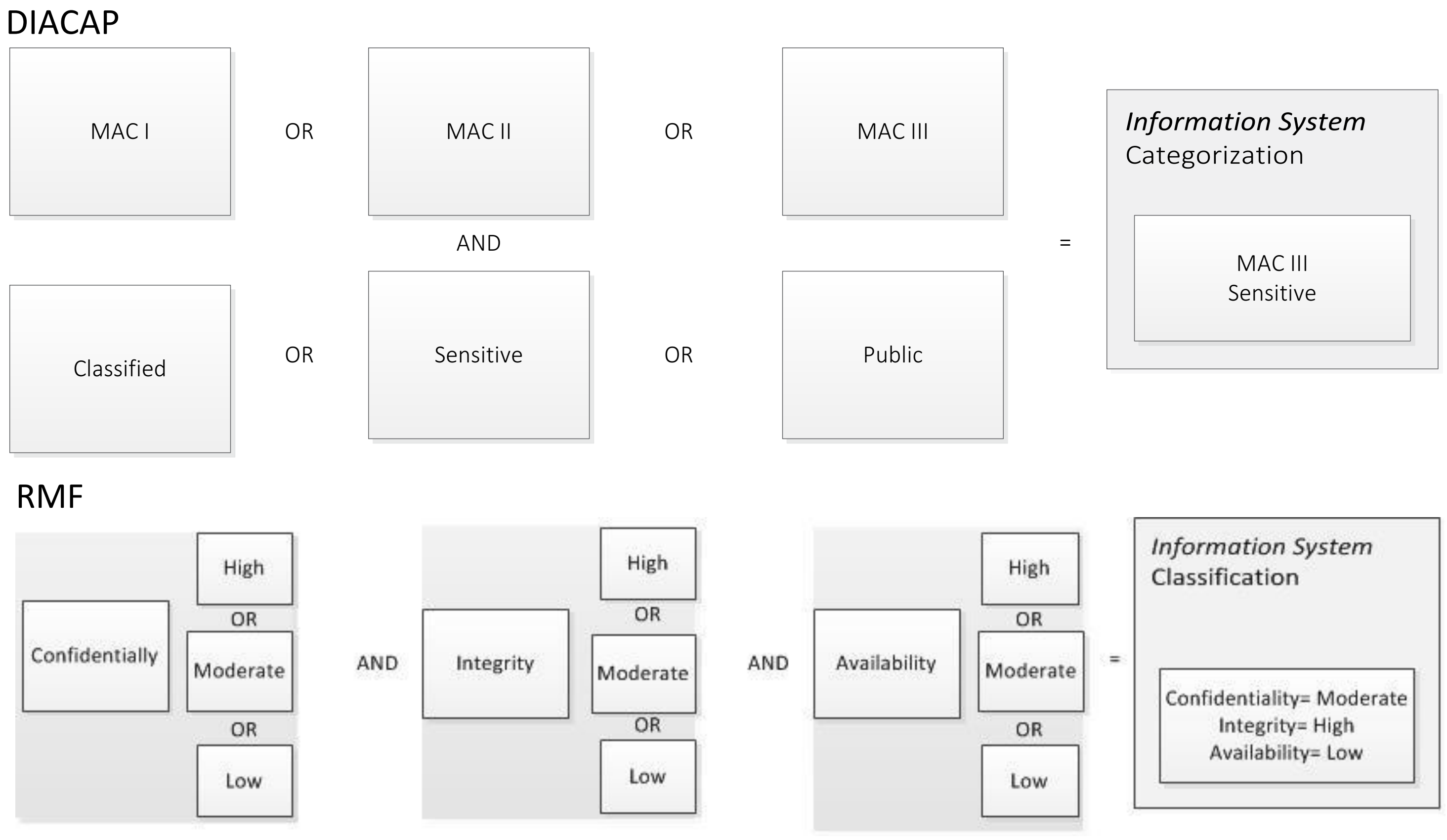
Security Controls

Security Controls, the cornerstone of any Cybersecurity program, conform to a new set of features and requirements for the RMF. Similar to the function of DoDI 8500.2 under DIACAP, the security control descriptions under the RMF are found in NIST Special Publication (SP) 800-53 (at time of this writing the publication was under Revision 4). The security controls within the publications that an IS is required to adhere to depends on the system categorization.

The process for determining an IS's Categorization has changed under RMF. DIACAP uses Mission Assurance Category levels (MAC I, II, III) to define the requirements for availability and integrity. The Classification Level (Classified, Sensitive, or Public) determines the confidentiality requirements. The combination of one MAC level AND one Classification level results in the IS's Categorization (i.e. MAC III, Sensitive). The RMF provides an evaluation of the three security objectives, Confidentiality, Integrity, and Availability individually and an impact level (Low, Moderate, or High) is assigned to each objective (i.e. Confidentiality= Moderate; Integrity= High; Availability= Low). The impact is based on what affect a realized threat will have on the system. The Committee on National Security Systems Instruction (CNSSI) No. 1253 directs the RMF system categorization.

DIACAP Role	RMF Role
DoD Chief Information Officer (CIO)	DoD Chief Information Officer (CIO)
Principal Accrediting Authority (PAA)	Principal Authorization Official (PAO)
DoD Component CIO	DoD Component CIO
Senior Information Assurance Officer (SIAO)	Senior Information Security Officer (SISO)
Principal Accrediting Authority (PAA) & Designated Accrediting Authority (DAA)[6]	Authorizing Official (AO)
Program Manager (PM)/ Systems Manager (SM)	Program Manager (PM)/ Systems Manager (SM) or Information System Owner (ISO)
Information Assurance Manager (IAM) & Information Assurance Officer (IAO)	Information System Security Manager (ISSM)
Information Assurance Manager (IAM) & Information Assurance Officer (IAO)	Information System Security Officer (ISSO)
Certifying Authority (CA) & Validator	Security Control Assessor (SCA)

Table 2: Security Roles Terminology Change



Figures 2a and 2b: DIACAP and RMF System Categorization

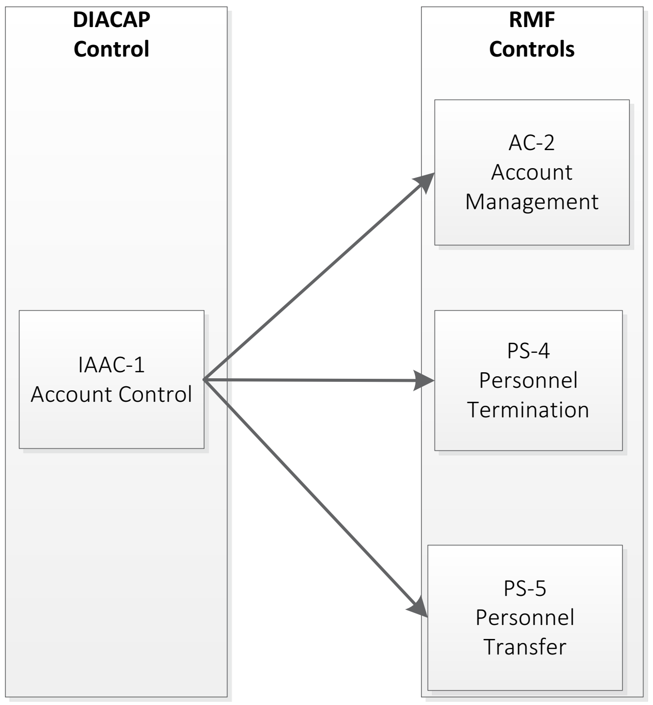


Figure 3: Example of control requirement granularity change from DIACAP to RMF

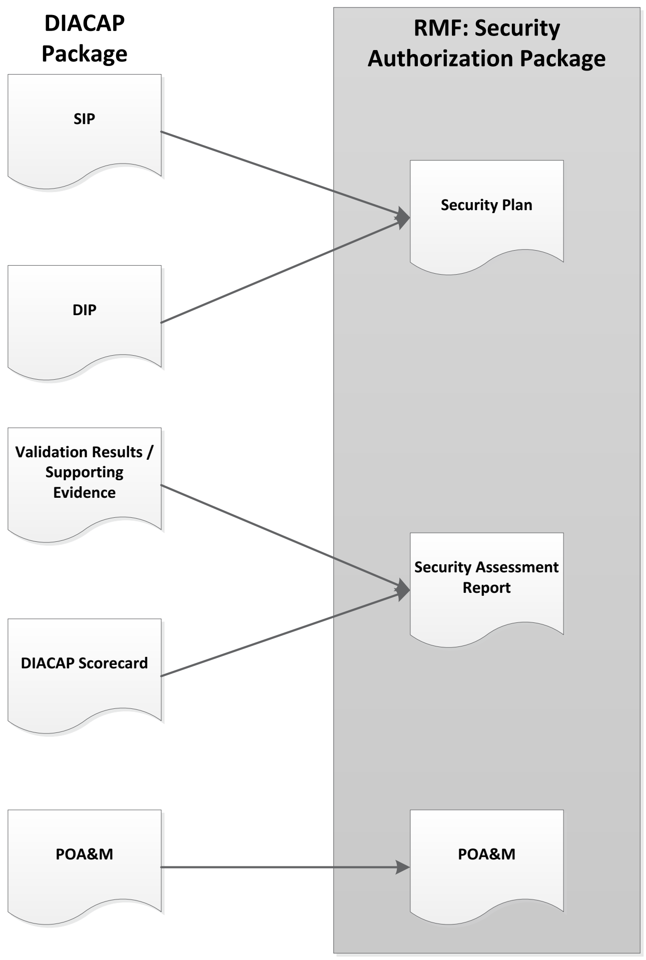

DIACAP Package
RMF: Security Authorization Package
SIP
Security Plan
DIP
Validation Results / Supporting Evidence
Security Assessment Report
DIACAP Scorecard
POA&M
POA&M

Figure 4: Artifact Transition DIACAP to RMF

A pronounced distinction between the DoDI 8500.2 catalog and NIST SP 800-53 is that it defines controls to mitigate risk in more detail. As a result, the IS's under the RMF have more controls required in order to meet the more well defined security requirements. In many cases the IS's could require triple the amount of controls under the RMF methodology. For example, the security requirements covered in DIACAP control, "Account Control" (IAAC-1), maps to multiple 800-53 controls, "Account Management" (AC-2), "Personnel Termination" (PS-4), and "Personnel Transfer" PS-5, as shown in Figure 3.

Although the number of required controls increases under RMF, because they are written at a more granular level, that does not signify an increased workload. The reality is that the overall security requirements are consistent between the two frameworks.

Artifacts

RMF reduces the artifact generation and submission process by removing the need for two separate package submissions. Under the RMF, artifacts have been streamlined leaving only one package per IS (not a Comprehensive and Executive package as with DIACAP). The three required artifacts under the RMF are the Security Plan, Security Assessment Report, and the Plan of Action and Milestones (POA&M). The relationship between the DIACAP Package artifacts and the RMF Security Authorization Package artifacts is illustrated in Figure 4.

Note that under the DIACAP model, while not required, it was common for an organization to have a formalized Security Plan at the discretion of the ISSM/ISSO. For the Cybersecurity teams developing a program under the RMF, the Security Plan is the cornerstone artifact in the program.

The Security Plan[7] provides an overview of the system, its security requirements and details the security controls in place.

"The fact that the Security Plan is the cornerstone of the RMF effort is an improvement over the DIACAP model. We needed to streamline this process and will need to evaluate how well the RMF works over time to see if we have it right. Just like we need continuous monitoring of our security efforts, we also need a periodic evaluation of our processes to ensure they are simple, understandable and executable. The security of our data systems is a daily battle that requires agile processes to meet the ever-changing cybersecurity demands." -Colonel Gregory T Breazile (Director, Cyber & Electronic Warfare Integration Division)

Continuous Monitoring

A component within the Security Plan receiving a new emphasis under the RMF is the Continuous Monitoring Strategy (CMS). CMS provides system-level strategy for evaluating the effectiveness of security controls and the observing of any changes to the system and environment. The strategy includes a plan for the annual assessments of implemented security controls.

The "assessor" must be independent of the IS requiring an external party to the organization not affiliated with either the control design or control execution. Other control elements implemented under the CMS may vary depending on the risk factors of the IS and the discretion of the ISSM.

Figure 5, illustrates three example elements of a CMS. Executing the CMS becomes critical under the RMF between the Authority to Operate (ATO) granted and expiration dates.

Along with the Security Plan, the CMS will be scrutinized and approved by the AO prior to proceeding further with the RMF. This new scrutiny, early in the RMF, further emphasizes the enhanced focus of the organization's continuous monitoring processes and the importance of identifying and coordinating resources needed to adequately execute the CMS.

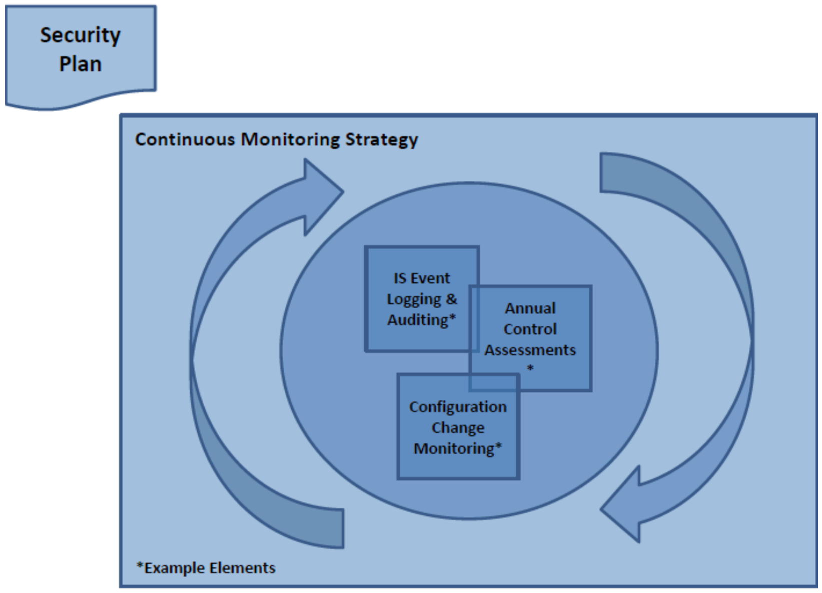
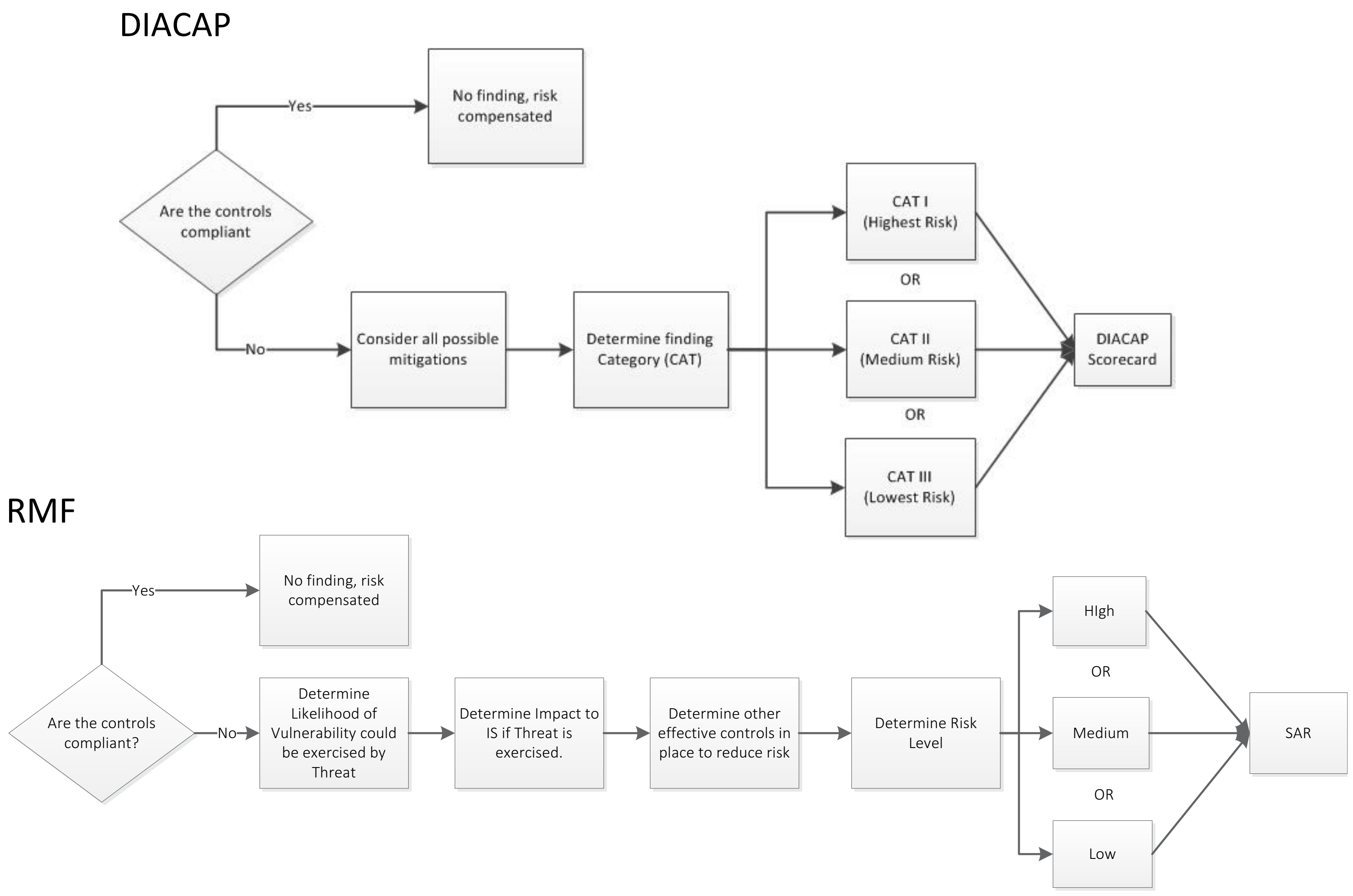

Security Plan
Continuous Monitoring Strategy
IS Event Logging & Auditing*
Annual Control Assessments*
Configuration Change Monitoring*
*Example Elements

Figure 5: Continuous Monitoring Strategy with Example Elements.

Security Assessment Report and POA&M

The Security Control Assessor (SCA) develops a plan for executing the Security Assessment, in order to populate the Security Assessment Report. The Security Assessor's role and the security assessment serve the same purposes as the Validator and validation process did within DIACAP. As in DIACAP every non-compliant control will have an associated risk level.

The DIACAP risk Categories (CAT I, CAT II, and CAT III) have been replaced in the RMF with the Security Assessor's evaluation of several factors determining the risk level. The risk level factor determination includes an analysis of the vulnerabilities caused by non-compliant controls and the threats that could exploit the vulnerabilities. Figure 6 presents the evaluation of non-compliant controls, different risk designations between DIACAP and RMF, and where these risk designations are recorded.


DIACAP
Are the controls compliant
Yes
No finding, risk compensated
No
Consider all possible mitigations
Determine finding Category (CAT)
CAT I (Highest Risk)
OR
CAT II (Medium Risk)
OR
CAT III (Lowest Risk)
DIACAP Scorecard
RMF
Are the controls compliant?
Yes
No finding, risk compensated
No
Determine Likelihood of Vulnerability could be exercised by Threat
Determine Impact to IS if Threat is exercised.
Determine other effective controls in place to reduce risk
Determine Risk Level
HIgh
OR
Medium
OR
Low
SAR

Figures 6a and 6b: Non-compliant risk determinations

The Validator captures non-compliant controls and risk determinations in the DIACAP Scorecard. Conversely, the SCA documents these results within the RMF's Security Assessment Report (SAR). Both the DIACAP Scorecard and the RMF SAR include an assessment of the overall system level of risk as well and both are required artifacts for an ATO decision.

In the same manner as the Test Plan findings in DIACAP, any non-compliant controls from the RMF's SAR carry in to the POA&M. The POA&M is a key artifact in the authorization package and the submitter maintains it throughout the system lifecycle.

Authorization Decision

The ISSM submits the Security Authorization Package, containing the Security Plan, SAR, and POA&M, to the AO for an authorization decision only when all three of these artifacts are complete. Figure 7 shows the logical progression of these artifacts, highlighting that the POA&M cannot be generated without the SAR which is dependent on the Security Plan.

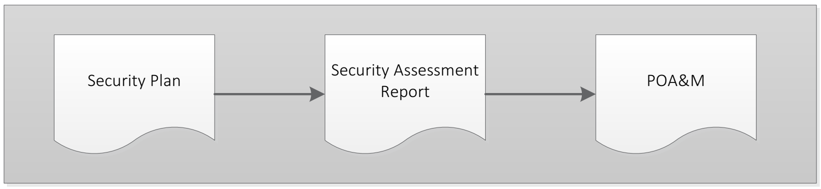

$$\begin{tikzcd}\text{Security Plan} \arrow[r] & \text{Security Assessment Report} \arrow[r] & \text{POA\&M}\end{tikzcd}$$

Figure 7: RMF Security Authorization Package Contents

Upon review by the AO, the authorization decision is codified as an Authorization To Operate (ATO), an Interim Authorization to Test (IATT), or a Denial of Authorization to Operate (DATO). IATTs should only be granted when an operational environment or live data is required to complete specific test objectives. IATT should normally expire in 90 days. Unlike DIACAP, RMF does not technically allow for an Interim Authority to Operation (IATO). RMF relies on the convention of issuing an "ATO with conditions" which must be met within a defined period of time. If those conditions are not met the AO may issue a DATO.

Reciprocity

An important design of the RMF is to improve efficiencies through reciprocity. Although the DoD branches followed common processes under DIACAP, the reissuance of DoDI 8510.01 for RMF provides explicit guidance on "reciprocity" that was formerly not as clear. Specifically, the guidance addresses coordination between deploying ISOs and PMs with receiving ISOs and PMs throughout the system development and the process for a receiving organization to accept an ATO. Ultimately, reciprocity increases transparency ensuring that AOs are equipped to make better informed decisions when accepting an existing ATO.

The transition to RMF enables reciprocity between the DoD and other Federal agencies. As stated above, the RMF will adhere to the security requirements under NIST 800-53 which is used as the Federal Government's common guidance for implementing security controls.

Software Engineering Institute | Carnegie Mellon University

IEEE

IEEE Computer Society | Software Engineering Institute

Watts S. Humphrey
Software Process Achievement Award

Nomination Deadline: October 15, 2015

Do you know a person or team that deserves recognition for their process-improvement activities?

The IEEE Computer Society/Software Engineering Institute Watts S. Humphrey Software Process Achievement Award is presented to recognize outstanding achievements in improving the ability of an organization to create and evolve software.

The award may be presented to an individual or a group, and the achievements can be the result of any type of process improvement activity.

To nominate an individual or group for a Humphrey SPA Award, please visit http://www.computer.org/web/awards/humphrey-spa

Conclusion

The transition to the RMF allows the Marine Corps to adopt a framework that dynamically responds to changes in risk. The RMF aligns itself with NIST publications that remain current in the face of emerging technologies. Ultimately, the RMF gives the Marine Corps a Cybersecurity program that is better designed to support the evolving Information Technology landscape.

Disclaimer

The views expressed are of the authors and do not represent any official position within the Department of Defense or the United States Marine Corps.

Acknowledgement

The authors would like to acknowledge LtCol Jeffrey Hammond (USMC), LtCol Michael Cho (USMC, Ret.), LtCol Floyd Means (USMC, Ret.) Marine Corps Information Technology Center Site Director, Captain Richard Wolferd (USMC) and Mr. James Klanke (President Global Project Management Group, Ltd.), and Dr. Jim Lee (Deputy Cyber Engineering, Marine Corps Systems Command) for their constructive criticism, comment, and review. Any errors remain the responsibility of the authors.

NOTES

1. Though written in the context of the DoD's adoption of RMF, the authors day to day work interactions are in direct support of the USMC and the nuances in this article may reflect or be biased toward that relationship.
2. Cybersecurity as opposed to Cyber Security is the parlance found in DoDI 8500.01; both terms are used interchangeably in many of the resources we reviewed.
3. Incorporates and cancels DoDI 8500.02, DoDD C-5200.19, DoDI 8552.01, et al.
4. This is a corrected diagram. The original reviewed for this paper shows DoDI 8500.02 as a publication applicable to RMF. It should also be noted that CNSSI 1253 is dependent on NIST 800-53, however under RMF, CNSSI 1253 guidance must be evaluated first prior to utilizing NIST 800-53.
5. At the time this article was written, the RMF Knowledge Service Website was still under development. Proposed URL is <https://rmfks.osd.mil>
6. The Marine Corps has already adopted the AO, ISSM and ISSO roles rather than using the DoD DIACAP terminology of PAA/DAA, IAM and IAO, respectively. The intention of this table is to be consistent with the DoDI for both DIACAP and RMF as a specific directive from the Marine Corps for RMF has yet to be released.
7. The RMF 'Security Plan' acts as a "road map" that guides reviewers to other important risk management and security design procedures such as the risk assessment, privacy impact assessment, system interconnection agreements, contingency plan, configuration management plan, and incident response plan. Once established, the Security Plan continues to be a dynamic document updated as needed to remain current, presenting an accurate picture of the ever evolving risk within the environment.

ABOUT THE AUTHORS

Major Henry R. Salmans III (USMC, Retired) of CSC is a former 4002/0602 Data Systems Officer/Communications Officer. His award winning work includes From Technological Triage To Maturing A Collaborative Environment (DoD International Command & Control Research and Technology Symposium), The American Way of War (War On The Rocks) and is an occasional guest writer for Ranger Up and the infamous Rhino Den. Currently, he advises the Technology Services Organization and the Cybersecurity Council of the Marine Corps Information Technology Center in Kansas City, Missouri.
Phone: 785-840-7066
Email: henryrsalmansiii@gmail.com

Mr. Andrew C. Tebbe, formerly of COmputing TechnologieS, Inc. (CoTs), is a civilian cybersecurity professional with the Marine Corps Installation Command (MCICOM) in Kansas City, Missouri, specializing in cybersecurity compliance and control assessment. Prior to joining MCICOM, he worked as an internal auditor for the USDA focusing on FISMA and FedRAMP compliance. As an IT security control auditor and consultant, Mr. Tebbe's private sector experience was with the public accounting firm KPMG LLP, the U.S. member of the International Cooperative.
Phone: 816-541-8848
Email: andrew.tebbe@mcw.usmc.mil

Mr. William J. Witbrod of COmputing TechnologieS, Inc. (CoTs) is a Fully Qualified Navy & Marine Corps Validator working for Installations & Logistics, Headquarters Marine Corps, for the Marine Corps Installation Command, Facilities Systems Branch in Kansas City, Missouri. Prior to joining CoTs in support of CSC, William served in the United States Army Signal Corps and held various executive security and audit positions in both the government and private sectors.
Phone: 913-244-4600
Email: witbrod@gmail.com

Model-Based Engineering for Supply Chain Risk Management

Dan Shoemaker, Ph.D., University of Detroit Mercy
Carol Woody, Ph.D., Software Engineering Institute

Abstract. Expanded use of commercial components has increased the complexity of system assurance verification. Model-based engineering (MBE) offers a means to design, develop, analyze, and maintain a complex system architecture. Architecture Analysis & Design Language (AADL), which has tools for modeling and compliance verification, provides an effective capability to model and describe all component units in a sourced product and implement practical measures for their management and assurance throughout the acquisition life cycle.

Supply Chain Context

The manner in which we develop complex software and systems has changed considerably since the dawn of the Internet age [1, 2]. Much of that change has occurred because software has become big business. The most recent estimate of the size of the industry is $407.3 billion per annum, with a 4.8% annual growth [3]. As a result of the vast increase in the scope of the marketplace, commercial components are ubiquitous in our infrastructure [4]. This is especially true in government where the Clinger-Cohen Act has directed federal agencies to maximize their use of commercial-off-the-shelf (COTS) products [5].

COTS applications, especially those that are developed for commercial purposes, are often vulnerable to exploitation [5]. This is especially true when the unique risks faced by government systems are factored in [5]. For instance, the conventional approach in most organizations is to look for a COTS product[1] to solve some functional need. That choice is understandable, due to the cost advantage and availability that COTS components represent [3]. However, implementing these products can represent a serious security challenge since we rarely have the option to fully understand what we are buying [6].

More importantly COTS products are typically integrated up a sourced supply chain, which creates a problem of security assurance and control at every level. It is a well-documented fact that we have lost all visibility into what is going on at the bottom of that generic chain [7]. So we are left with "trust" as the only viable option for establishing assurance. But what evidence can we use to justify the trust? Since market forces favor functionality over security and reliability, the challenges of addressing this supply chain problem are increasing [6].

It would be valuable to have a formal, well-defined, standard and systematic means for evaluating the assurance of systems which may contain security vulnerabilities inserted through insecure information and communications technology (ICT) supply chains. That formal solution is a practical necessity if we ever want to be assured that our adversaries cannot, "destroy power grids, water and sanitary services, induce mass flooding, release toxic/radioactive materials, or bankrupt any business by inserting malicious objects into the (ICT) components that comprise our infrastructure" [8].

The Problem: Why ICT Supply Chain Security is so Easy to Compromise

Because of the potentially critical impact of insecure supply chains on the U.S. infrastructure, the General Accounting Office has placed ICT Supply Chain Risk Management on its annual "Key Issues, High Risk" list [9]. A primary contributor to security vulnerabilities is the standard approach used for integration [5]. We no longer build single purpose systems from the ground up using a conventional design-build-test structure of creating well-defined components assembled into a predicted whole. Instead, we integrate a system from existing and reusable components in a hierarchy that extends up from the modular level through an increasingly sophisticated larger collection of integrated modules.

Due to the cost advantage the components that we integrate are obtained through a world-wide ICT supply chain that favors low cost [4]. In effect our products can be composed of software artifacts from India, chips and programmed logic from Korea and small components from Vietnam and China [4, 7]. These components were not designed and built to work together smoothly and effectively, instead they are cobbled together using standardized interfaces for information passing. These components were not built to only perform the selected activities needed for a specific system and may provide a wide range of additional functionality that supports unintended consequences.

The organization implementing this assembled collection of international components needs to ensure consistency in performing all aspects of supply chain risk management by providing a uniform, disciplined repeatable assessment that establishes an appropriate level of trust. One source of evidence would be a means of confirming that the practices involved in developing each level of the final product were consistently executed using a uniform standard management process starting from the basic components through to the final assemble within a planned and documented environment. Another source of evidence would be an assessment of "what could go wrong in the construction process (i.e., assessing risks), determining which risks to address (i.e., setting mitigation priorities), implementing actions to address high-priority risks and bringing those risks within tolerance must be well defined and uniform" [10].

Correct application of these control and assurance activities requires a detailed understanding of how the product will be built and the ability to monitor the construction process to assure security and correctness throughout. That is easier said than done since the basic component elements for a system are likely to be complex code segments that are sourced outside of the direct control of the system manufacturer, who is actually an integrator of a range of components from many sources.

Problem Statement: The Weakest Link

Typically, supply chains are hierarchical, with the primary supplier forming the root of a number of levels of parent-child relationships. From an assurance standpoint, what this requires is that every individual product of each individual node within that hierarchy be secure as well as correctly integrated with all other components up and down the production ladder. [7].

In the world of software, the locus of assurance is typically in the manufacturing process rather than the product itself. That is because the actual product is both too complex, as well as too virtual to be able to see and control. Ensuring a complex, distributed process like a supply chain would require a coordinated set of standard, consistently executed activities to enforce the requisite level of visibility and control. Yet, because the development process is usually occurring in a number of disconnected global locations, typically at the same time, the requisite level of understanding needed to assure the security and correctness of the component is hard to achieve [10].

For effective supply chain risk management, system engineering staff need to be able to evaluate the likelihood and impact of a potential vulnerability appearing at any stage in the product development. This evaluation mechanism is needed in order to identify appropriate mitigations. Best practice suggests that this identification and evaluation should take place during the early acquisition stages, because the form of the final product is being defined at that point [4, 5]. This challenge for the identification of potential security vulnerabilities extends the typical system engineering security considerations beyond the borders of the acquirer-supplier relationship. In that respect the correctness and security of the supply chain that comprises the development environment also impacts security [11]. To address complex supply chain risks, systems engineers need to be able to analyze the potential for the components to be exploited or subverted at any level of the construction process, and then consider the potential actions that need to be taken in order to detect exploitation and devise the necessary mitigations to continue to operate [11].

Systems engineers would want to have the ability to identify potential vulnerabilities early in the system-acquisition process. This would require that systems engineers be able to fully dictate the system concepts and critical functions and access paths of the entire product as it is being built. However, the typical approach in use today is to simply test for known common vulnerabilities of the system, supply chain, and development environment when all the pieces are assembled at integration. These tests are drawn from industry databases [1, 11], or in the case of the Federal Government from the Defense Acquisition Guidebook [12]. This is insufficient to evaluate and assign trust to the end product. It is desirable, in some manner, to assure each component at the bottom of the supply chain process and then every successive integration to the final product. While we may not have the mechanisms to fully address this need, we can greatly improve current practice.

Utilizing Formal Structured Design Approaches to Establish Assurance Control

The identification and detailed understanding of all of the outsourced units in a supply chain can be an impossible task where the product might be composed of 10,000 individual components at the 4th or 5th level down in the integration process.

The logical way to draw up such a schematic is through a formal system modeling process, which will uniquely identify each component at all levels in the hierarchy. Such a modeling approach, if properly supported by a formal design language and kept under configuration management control, could create a very precisely detailed representation of the functional and security requirements for all components. And that schematic model would allow for the rational imposition of structured assurance activities at all levels in the integration process.

The acquirer must establish specific properties that the integrators must meet in order to satisfy the acquisition contract. If the system engineers for the acquirer assemble the first level decomposition of the system into AADL and incorporate the required properties that each component must contribute to the system into the model, the planned composition can be formally verified using available tools. The developer of each component must deliver an AADL model that establishes the properties of their component as built along with the actual product. In addition to executing acceptance testing of the component, the acquiring engineer can import this decomposition into the original model and confirm that the expected properties are still met. The result, from an assurance control standpoint, is the assembly of a complete and explicitly detailed set of design schematics that can be used to guide the process of monitoring the award and assurance of the outsourced work.

Safety-critical verification of cyber-physical systems (CPS) has benefited from the use of architecture fault modeling capabilities provided by Architecture Analysis & Design Language (AADL). Architecture led hazard analysis using architecture description languages (ADLs) such as AADL has become an effective capability in safety fault management. The cost of successfully addressing safety compliance has been greatly reduced through the use of extensions to AADL that automate safety analysis and produce safety assessment reports to meet recommended practice standards (such as SAE ARP4761) [16]. AADL is a relatively well known architectural description language that was first developed as the Avionics Architecture Description Language [14]. It was eventually standardized by the Society of Automotive Engineers as AADL [14]. The Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) has used AADL to effectively address design verification for the qualities of safety, reliability, and performance [18, 19]. Other researchers have incorporated selected attack scenarios into modeling languages (e.g., OCL [20], OWL-DL and SWRL [21], and SysML [22]). Avionics vendors have successfully used formal modeling to ensure safety properties from components developed by multiple vendors.

Acquirers can leverage these successful capabilities to address security as well as safety in supply chain risk management. The AADL modeling process ensures all three of the requisite criteria for successful management of a system product under development. These are, understanding, through modeling, assurance through formal validation and verification and finally accurate control of the integration of components, through reference to well-defined baseline expected behaviors.

Understanding: it is essential to have a detailed description of the structural elements of the system architecture in order to enforce assurance. The major problem with ensuring trustworthy product security at the component level using standard systems engineering approaches is the inability to precisely document the behavior of the specific contents of secured products along with the exact process by which they are built. Consequently, it would seem impossible to verify that the design requirements and tenets of best practice have been satisfied for each unit, especially if the system code has already been written, which is the case with commercial products. However, expected operational properties can be described and evaluated across the composition. The behaviors for each component can be formally described using a modeling language that captures the detail evidence. AADL is supported by tools that can capture properties about each component as well as the selected integration mechanisms for each component into the composition. Assumptions can be checked for consistency across all components to meet critical system behaviors.

Assurance: The necessary level of visibility for assurance requires knowing what the code that comprises each component is supposed to do and how that will affect the assembled whole. The confirmation of correctness of a complex system is normally presented in the form of an assurance case [11]. An assurance case is a well-thought-out argument, which is often represented notionally. An assurance case proposes and supports whatever claims are made about a given set of planned behaviors for a system [11]. An assurance case can be structured as a proof of trustworthiness based on evidence assembled through a formal model.

Homeland Security

The Department of Homeland Security, Office of Cybersecurity and Communications (CS&C) is responsible for enhancing the security, resiliency, and reliability of the Nation's cyber and communications infrastructure and actively engages the public and private sectors as well as international partners to prepare for, prevent, and respond to catastrophic incidents that could degrade or overwhelm these strategic assets. CS&C seeks dynamic individuals to fill critical positions in:

- Cyber Incident Response
- Cyber Risk and Strategic Analysis
- Networks and Systems Engineering
- Computer & Electronic Engineering
- Digital Forensics
- Telecommunications Assurance
- Program Management and Analysis
- Vulnerability Detection and Assessment

To learn more about the DHS, Office of Cybersecurity and Communications, go to www.dhs.gov/cybercareers. To apply for a vacant position please go to www.usajobs.gov or visit us at www.DHS.gov.

The supporting evidence for an assurance case may combine different kinds of documentation and data to justify its generic claim [11]. So a structured model of the system architecture, down to the unit level, is needed to let the engineer confirm the correctness of each component in the system [11]. In that respect, the assurance claim starts with the development of a general statement about a system goal such as an operational timing requirement. It is then decomposed down to a detailed, structured system model, which provides the concrete description needed to evaluate the detailed system architecture and its security properties at all levels in the development process as well as up and down the supply chain [11]. Claims for supply chain assurance can also be decomposed into an assurance case that assigns various desired (and necessarily avoided) characteristics to each component.

Control: In most approaches for systems engineering, modeling and validating of the attributes of a system are typically done using existing threat modeling and analysis technologies which are separate from the actual components and known characteristics of the system. The focus is on the creation of threat models to inform requirements and development decisions instead of formally evaluating the system composition to see how well it addresses the needed assurance. This provides no mechanisms for verification of delivered results to determine if threat concerns have been appropriately addressed. Moreover, these models are typically not maintained or updated throughout the life cycle, making it difficult to predict the impact of downstream change on attributes that cut across unit boundaries within the system [13]. AADL provides a means to model expected system behaviors that address threat concerns and formally verify consistency of these behaviors in modelled components that are outsourced.

The impact of component decomposition choices, which can include COTS, open source, and other outsourced subcontractors, need to become the responsibility of the integrator assigned to deliver the component to prove that desired system properties are met. The model of their resulting product must include downstream alterations made during the coding and even testing stages that are likely to produce discrepancies between the original design and its assurance case [13]. If the lower level components can embody undesired and unanticipated emergent behavior that can propagate up the ladder as the system evolves to higher levels of integration [11] this can be identified if the product behavior is appropriately modeled. Modeled properties must be applied to both component development and their interconnections.

AADL implements a basic language construct that centers on use of a standardized notation. The existence of a single common notation makes AADL relatively easy to automate since it underwrites a single standard view of all aspects of the system [14]. The notation language allows the engineer to specify system-specific characteristics by describing the unique set of user properties as well as any behavior involving state machines. It also allows the engineer to specify all of the associated error and dissemination concerns along with the specific data constraints [15].

The value of the AADL modeling scheme lies in the ability to isolate the basic components of a system to the right level of abstraction. This is called kernelling [4]. Kernelling allows for specific understanding and verification of the correctness of all components at a specified level of the design. The ability to describe discrete units within security levels in the architecture enables the practical work of validation and other forms of black-box testing. It is possible to consider each unit as an individual component of the proper functional level. Interactions of components both within that level and with objects at a higher level can be understood for the purposes of targeted testing of each individual component, or the testing of an integrated set of functions at whatever level designated.

AADL provides tools that capture the assurance case so that it can be verified against the formal design to identify inconsistencies. By keeping both the model and the assurance case current throughout the development life cycle and across the supply chain, this consistency checking can be used when COTS components are replaced with updated versions or other components, when new components are added that change the overall composition, and when segments of a system are modernized to meet changing business needs.

AADL is not a simplistic solution to supply chain risk management. To take advantage of this modeling capability, the system and its expected properties must be clearly established within the model prior to outsourcing. The assurance case must be assembled and verified against the system model to ensure the desired properties are consistently applied to the components. Security concerns expressed in a threat model must be analyzed and design mitigations characterized as expected properties within the model. Trust boundaries between components must be clearly described with properties that formalize allowed behaviors. Current SEI research is exploring the range of security behaviors that can be modeled to establish a level of effectiveness and the potential for code generation capabilities to carry formally defined behaviors directly into the resulting product.

Conclusion: Application of AADL to Assurance in the Supply Chain

The use of AADL to provide a model-based engineering approach (MBE) offers a better way to design, develop, analyze, and maintain a system architecture that is supported by a supply chain. AADL, provides an effective capability to model and describe all component units in a sourced product and implement practical measures for their management and assurance throughout the acquisition life cycle [14, 15]. Modeling enhances the engineer's ability to identify and address potential design weaknesses, an important category of security problems as noted in the common weakness enumerations [23].

Through the application of MBE system engineers, architects and product developers can reduce risk by performing early and repeated analysis of the system architecture [14] as outsourced components are delivered. This level of control and assurance can reduce cost by ensuring fewer system integration problems as the product moves up the supply chain.

A formal model can also simplify and ensure more effective evaluations of the organization-wide impacts of architectural choices and, by increasing understanding, make for simpler life-cycle support [14]. MBE can also increase confidence in the security of the implemented product because the assumptions that are captured in the modeling can be shown to have been validated as the product moves up to implementation in the operational system [14].

Formal modeling of all system components at all levels of decomposition gives the engineer the opportunity to define the detailed security characteristics of a delivered sourced product. Thus the MBE approach based on AADL design descriptions should provide the potential for use in the overall verification and validation work through confirmation of an assurance case to confirm the correctness of all components in a product [14, 15]. AADL modeling can be used to evaluate the correctness of components as well as the integration of components provided through the supply chain from multiple suppliers [13].

AADL has been successfully utilized to model both software and hardware applications in industries where the need for safety and reliability is paramount [14, 15]. SEI research is working to expand the security analysis capabilities of AADL [14, 15] to incorporate the design characteristics and constraints needed in formal modeling to anticipate and avoid two critical security design challenges: acceptance of tainted input and allowing inappropriate elevation of privileges. AADL offers opportunities for improvement in supply chain risk management.

Acknowledgment

Copyright 2015 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States 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.

This material has been approved for public release and unlimited distribution.

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University.

DM-0002424

NOTE

1. Many of the same challenges apply to open source

ABOUT THE AUTHORS

Dr. Carol Woody has been a senior member of the technical staff at the Software Engineering Institute, Carnegie Mellon University since 2001. Currently she is the technical lead of the cyber security engineering team whose research focuses on building capabilities in defining, acquiring, developing, measuring, managing, and sustaining secure software for highly complex networked systems as well as systems of systems.

Daniel P Shoemaker, PhD, Principal Investigator and Senior Research Scientist at UDM's Center for Cyber Security and Intelligence Studies. This Center includes the Computer Information Systems-Information Assurance Department, as well as the Center of Academic Excellence for National Security Agency. The Center has just completed a two-year Department of Defense Contract to develop Software Assurance Curriculum and Courseware. Dan is a full time Professor at University of Detroit Mercy with 25 of those years as Department Chair. As the Co-Chair for the, National Workforce Training and Education Initiative he is one of the Authors of the National Software Assurance Common Body of Knowledge (CBK) for the Department of Homeland Security. But while Dan spends a lot of time in DC, he is a Michigan man at heart, beginning with his education at the University of Michigan and the outreach opportunities he shepherds within the State of Michigan through his leadership of the International Cyber-Security Education Coalition. This Coalition covers a five state region with research partners as far away as the United Kingdom. Dan also spends his free time authoring some of the leading book in Cyber Security. Look for his newest edition to hit the press, this spring Cyber Security: The Essential Body of Knowledge, based on the DHS National Cyber Security Division's EBOK. His last book, Information Assurance for the Enterprise, is McGraw-Hill's primary textbook in that field and was number one on Amazon's list for three consecutive years. His next book Engineering a More Secure Software Organization, which is also published by Cengage, will be out next spring.

REFERENCES

1. MacDonald, Neil; Valdes, Ray Gartner Says IT Supply Chain Integrity Will Be Identified as a Top Three Security-Related Concern by Global 2000 IT Leaders by 2017. Gartner 2012
2. Mitre Corporation. 2012. Common Weakness Enumeration: A Community-Developed Dictionary of Software Weakness Types. <http://cwe.mitre.org>
3. Pettey, Christy, "Gartner Says Worldwide Software Market Grew 4.8 Percent in 2013" Gartner, Stamford, Connecticut, March 31, 2014
4. Reifer, Donald J. "Malicious Code in COTS: A Quantitative Study", Reifer Consultants, Inc. June 2007
5. Baldwin, K., J. F. Miller, P. R. Popick, and J. Goodnight. "The United States Department of Defense Revitalization of System Security Engineering through Program Protection." Proceedings IEEE Systems Conference (SysCon), Vancouver, CA-BC, March 2012
6. Wilshusen, Gregory C., "IT SUPPLY CHAIN: Additional Efforts Needed by National Security-Related Agencies to Address Risks." United States Government Accountability Office, Testimony before the Subcommittee on Oversight and Investigations, Committee on Energy and Commerce, House of Representatives, March 27, 2012
7. Evans, G., "Flying fraudulently – How a Weak Supply Chain Became the USAF's worst Enemy," 2012, URL: <http://www.airforce-technology.com/features/feature-supply-chain-us-air-force-fraudulent-parts> [Accessed April, 2015].
8. Clark, R.A. and Schmidt, H.A., "National Strategy to Secure Cyberspace, Washington, D.C.", The President's Critical Infrastructure Protection Board, 2003
9. United States Government Accountability Office, "HIGH-RISK SERIES An Update", Report to Congressional Committees, February 2015
10. John Steven, "Adopting an Enterprise Software Security Framework", IEEE Security & Privacy, vol.4, no. 2, pp. 84-87, March/April 2006
11. SEI (Software Engineering Institute). 2009. CMU/SEI-2009-TR-010. SEI Secure Design Patterns. Pittsburg, US-PA: Carnegie-Mellon University, Software Engineering Institute.
12. United States Department of Defense, Defense Acquisition Guidebook, Chapter 4 – Systems Engineering, Production Date: 15 MAY 2013
13. Woody, Carol, Robert Ellison and William Nichols, "Predicting Software Assurance Using Quality and Reliability Measures" Technical Note, CMU/SEI-2014-TN-026 CERT Division/SSD, December 2014
14. Software Engineering Institute, Architecture Analysis and Design Language, <http://www.sei.cmu.edu/architecture/research/model-based-engineering/aadl.cfm>, accessed April 2015
15. Gacek, Andrew, John Backes, Darren Cofer, Konrad Slind and Mike Whalen, "Resolute: An Assurance Case Language for Architecture Models," Cornell University, arXiv: 1409.4629v1 [cs.SE] 16, Sep 2014
16. Cervin, A. & Lund U., Impact of Scheduler Choice on Controller Stability, CCACSD 2006
17. Firesmith, D., Common Concepts Underlying Safety, Security, and Survivability Engineering, CMU/SEI-2003-TN-033, <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6553>
18. Bruce Lewis, Jérôme Hugues, Lutz Wrage, Peter Feiler, John Morley, "Model-Based Verification of Security and Non-Functional Behavior using AADL", IEEE Security & Privacy, 2009
19. Julien Delange Wheel Brake System Example using AADL; Feiler, Peter; Hansson, Jörgen; de Niz, Dionisio; & Wrage, Lutz. System Architecture Virtual Integration: An Industrial Case Study (CMU/SEI-2009-TR-017). Software Engineering Institute, Carnegie Mellon University, 2009 <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9145>
20. Almorsy, M.; Grundy, J.; Ibrahim, AS., Automated software architecture security risk analysis using formalized signatures, Software Engineering (ICSE), 2013, pp.662,671, 18-26 May 2013
21. Asnar, Y., Paja, E., Mylopoulos, J., Modeling design patterns with description logics: A case study (2011) Lecture Notes in Computer Science, 6741 LNCS, pp. 169-183.
22. Ouchani, S., Jarraya, Y., Ait Mohamed, O., Model-based systems security quantification (2011) 2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011, art. No. 5971976, pp. 142-149.
23. Mitre, Common Weakness Enumerations, <http://cwe.mitre.org/>

NASA's Approach to Software Assurance

Martha Wetherholt, NASA

Abstract. NASA defines software assurance as: the planned and systematic set of activities that ensure conformance of software life cycle processes and products to requirements, standards, and procedures via quality, safety, reliability, and independent verification and validation. NASA's implementation of this approach to the quality, safety, reliability, security and verification and validation of software is brought together in one discipline, software assurance. Organizationally, NASA has software assurance at each NASA center, a Software Assurance Manager at NASA Headquarters, a Software Assurance Technical Fellow (currently the same person as the SA Manager), and an Independent Verification and Validation Organization with its own facility. An umbrella risk mitigation strategy for safety and mission success assurance of NASA's software, software assurance covers a wide area and is better structured to address the dynamic changes in how software is developed, used, and managed, as well as it's increasingly complex functionality. Being flexible, risk based, and prepared for challenges in software at NASA is essential, especially as much of our software is unique for each mission.

Background

NASA's system safety and system reliability, has traditionally looked at the software of the system as either "it works" or "it does not work." Not that NASA did not do good software development and develop extensive software fault tolerance approaches, but NASA relied on the hardware for the safety aspects even as software took on more complex and critical functions and most of the roles for fault detection, isolation and recovery. The amount of software in our earliest missions was relatively small and straight forward compared to the missions of today. Originally, Space shuttle and then the International Space Station worked with balancing hardware's safety role with strict development and design criteria for any software that was considered "safety critical." The software safety criteria coming from early NASA projects was so strict, that many projects tried to avoid having their software labeled as safety critical. The system safety teams on many projects often did not have personnel with the software expertise for examining software at the appropriate level. At that time, those system safety teams with limited software resources and not directly part of the Shuttle or ISS, seldom were not able to go much lower than considering software as a system component that either did or did not work. These teams with limited expertise were unable to take into account the many ways software can fail, let alone why and what the impacts were on the system. NASA, for most of those earlier projects, relied on the hardware. Shuttle and ISS recognized the need and created a special computer software safety committee to review software issues and support the program safety panel(s) (which review the hazard analyses processes for projects from start to acceptance) and help projects when software becomes critical. As software evolved, taking on more and more functionality with growing system complexities, NASA saw the need for software assurance to grow as well. While many felt that providing software process checks and software product evaluations was sufficient, the reliability and safety aspects of software was, and in some cases still is, undervalued. In the 1990's, software safety at NASA was further promoted via an agency standard and guidebook that were produced to lay out the principles of both analyzing the software for contributions to system faults and failures as well as assessing the risk software takes on in reporting and mitigating hardware and system hazards. The lessons learned from the Shuttle software safety processes were incorporated and analyses and evaluation methods were stressed as well as providing support for tailoring the safety effort to the project. Software Assurance and its other sub-disciplines have been growing and evolving as well.

Software Assurance

The software assurance process is the planned and systematic set of activities that ensure conformance of software life cycle processes and products to requirements, standards, and procedures. Software assurance assures that the software and its related products meet their specified requirements, conform to standards and regulations, are consistent, complete, correct, safe, secure and as reliable as warranted for the system and operating environment, and satisfying customer needs. Note, scientific principle investigators, many of our customers, sometimes need a continuing discussion to discover what is needed verses what is "wanted" and what is possible as we push forward the principles of science and physics. Thus, some requirements are actually "desirements" where something less is actually sufficient; and sometimes NASA can provide them with more than they knew was possible or alternative solutions.

Software assurance reviews and analyzes all processes used to acquire, develop, assure, operate and maintain the software independently; evaluating if those processes are appropriate, sufficient, planned, reviewed, and implemented according to an adequate plan, meeting any required standards, regulations, and quality requirements. Software assurance utilizes relevant project-based measurement data to monitor each product and process for possible improvements. NASA software assurance has begun to work with the NASA Chief Information Office and Protective Services Office to assess the role of software assurance for mission software security. It is a joint effort between Software Assurance, the Chief Engineers Office, Project and program management as well as the CIO and Protective services to address the many facets of mission development and operational environment security.

At NASA, Software Assurance has evolved in to an umbrella risk identification and mitigation strategy for safety and mission assurance of all NASA's software [Figure 1]. It provides a consistent, uniform basis for defining the requirements for software assurance programs to be applied and maintained throughout the life of that software, that is, from project conception, through acquisition, development, operations and maintenance, and then evaluates if the software is properly retired.

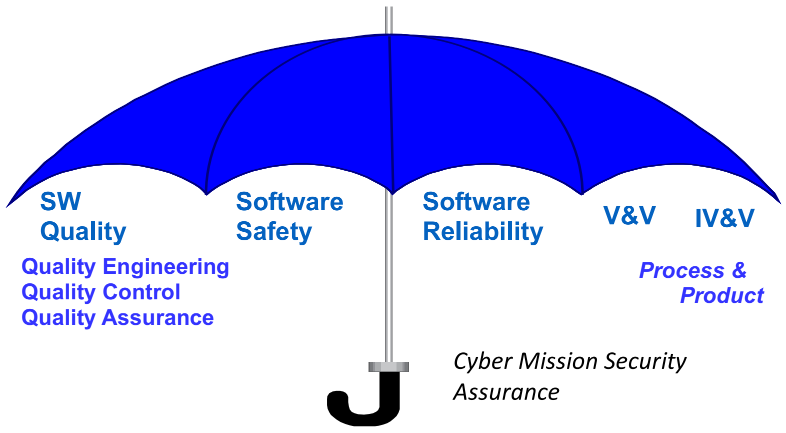
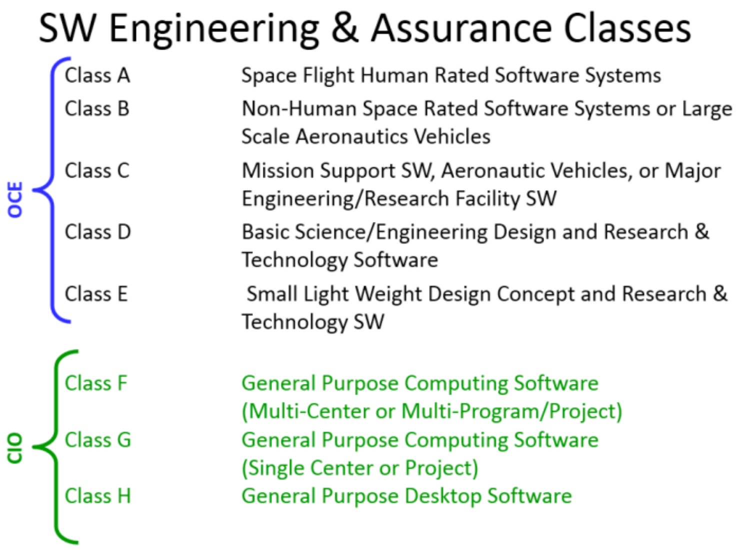


Figure 1. NASA's Software Assurance Umbrella of Risk Mitigation



SW Engineering & Assurance Classes

OCE	Class A	Space Flight Human Rated Software Systems
	Class B	Non-Human Space Rated Software Systems or Large Scale Aeronautics Vehicles
	Class C	Mission Support SW, Aeronautic Vehicles, or Major Engineering/Research Facility SW
	Class D	Basic Science/Engineering Design and Research & Technology Software
	Class E	Small Light Weight Design Concept and Research & Technology SW
CIO	Class F	General Purpose Computing Software (Multi-Center or Multi-Program/Project)
	Class G	General Purpose Computing Software (Single Center or Project)
	Class H	General Purpose Desktop Software

Table 1. NASA Software Classifications

The purpose of software assurance is to assure that software products are of sufficiently high quality and operate safely, securely and reliably. This includes products delivered to and used within NASA, and products developed and acquired by NASA. Software assurance assists in risk mitigation by helping expose potential defects in products and processes, thus preventing problems from evolving. However, it also, through its metrics, tracking and analyses activities, enables improvement of future products and services. Software assurance often serves as the corporate memory from project to project, sharing potential problem areas and lessons learned.

Software engineering and the software assurance disciplines are integrally related and yet each has its own responsibilities. Jointly they are responsible for providing project management with the optimal solution for software to meet the engineering, safety, quality, and reliability needs of the project. This necessitates a close working relationship to establish the appropriate levels of effort for both. The NASA Procedural Requirements, NPR 7150.2, NASA Software Requirements invokes the NASA Software Assurance Standard (NASA-STD-8739.8) and the NASA Software Safety Standard (NASA-STD-8719.13), requiring a close working relationship, understanding of roles and responsibilities, and establishing expected communication paths. NPR 7150.2, besides laying out the NASA minimum requirements for software development, provides the NASA software classification upon which software engineering, software assurance and software safety all base their tailoring. Table 1 shows the NASA Software Classes and a very brief, summary definition of them. The Office of the Chief Engineer "owns" Software Classes A-E as those are used for Mission software and support while the Chief Information Office "owns" the infrastructure software like desktop operating systems and applications, web based applications, etc. which are Software Classes F, G & H. Software Assurance has focused on the mission software.

The NASA Software Assurance and Safety standards are also invoked from the Agency System Safety, Reliability and Quality policies and procedures, thus stating not just the recognition of software assurance as an explicit special discipline, but also the expectation of software assurance as part of the joint assurance, safety and reliability support to NASA's systems. The struggle is balancing the need for software to be part of the overall system assurance, safety and reliability analyses and having the expertise needed to take those systems analyses down to the proper depth to see the potential impacts of software errors on that system.

The NASA Software Assurance Standard (NASA-STD-8739.8) provides a common framework for software assurance definition, activities, and implementation across NASA and its contractors. It provides tailoring recommendations in order for software assurance planning and execution to meet the needs of different flight, ground, facility and experimental software projects. The NASA Software Safety Standard lays out a systematic approach to software safety as an integral part of the overall systems safety, establishing the activities, data, and documentation necessary for the acquisition and development of software in a critical system. It also defines the levels of criticality for software, starting with a "litmus test" to determine if the software is safety critical (see list below) or not. Then provides additional risk based scoping based on severity and likelihood of occurrence, level of autonomy, complexity, and time to criticality. Time to criticality is important and changes with missions and functions, it is the amount of time to detect, recognize and react to a fault or potential failure before it becomes a failure, or if a failure occurs, then the time to put the system in to safe mode while correcting the problem. This determines the level and extent of autonomy of the fault detection, isolation and recovery activities. Both standards provide a clear acquirer –provider perspective as well as tailoring that meets the level of effort for the software class and criticality.

NASA software is classified as safety critical if it meets at least one of the following:

a. Causes or contributes to a system hazard/condition/event.
b. Provides control or mitigation for a system hazards/condition/event
   (1) Controls safety critical functions.
   (2) Mitigates damage if a hazard/condition/event occurs.
   (3) Detects, reports, and/or takes corrective action, if the system reaches a potentially hazardous state.
c. Processes safety critical commands (including autonomous commanding)
d. Resides on the same processor as safety critical software and is not logically separated from the safety critical software.
e. Processes data or analyzes trends that lead directly to safety decisions (e.g., determining when to turn power off to a wind tunnel to prevent system destruction).
f. Provides full or partial verification or validation of safety critical systems, including hardware or software subsystems. (e.g., this can include models and simulations)

With the basic software engineering and assurance requirements firmly established across NASA, it becomes a matter of training, implementation and improvement. NASA has a robust training program for software assurance. The NASA Safety Center maintains not only NASA created instructor and web-based training on all the sub-disciplines of software assurance, it also contracts with outside experts to bring in specialized training where needed. NASA's Software Engineering also has agency wide training that software assurance participates in as well as project level training for project specifics.

Organizationally, within NASA, the number of actual practitioners of software assurance assigned to the independent offices of Safety and Mission Assurance across the Agency may be relatively small. Thus, the requirements are intentionally written so that many different groups may perform different aspects of software assurance (e.g., systems engineering might perform the software safety analyses, software engineering might collect and trend defects). An entity/organization independent from the organization creating the software still is required to either perform or guarantee that software assurance activities are performed correctly and to the necessary level, and that records of those activities are created, analyzed, and maintained. Software Assurance metrics are also important. Software engineering and software assurance organizations share many software product quality metrics and process metrics, but NASA also requires software assurance performance metrics, to track and measure the performance of the software assurance activities and to improve activities for missions. Many software assurance activities may be tailored and performed within the project structure, but a group independent from the project evaluates those activities and the results. For NASA this is the Safety and Mission Assurance (SMA) organization; for a contractor, this should be a managerially separate safety and assurance organization which should be called out in the contract. Often, one or more software assurance engineers from an SMA organization may be assigned to work with a project throughout its life cycle. While these software assurance engineers are a part of the project and participate in day-to-day activities, perform most or all of the assurance functions, and attend project meetings and reviews, they maintain a separate reporting chain through their SMA organization. This activity is much like an oversight role, that is, the software assurance engineers are closely tied in with the project and provide input on a daily basis. At other times, the independent organization, SMA, may provide only insight for the project, evaluating if the software assurance activities are performed and performed sufficiently by the project personnel and participating more by audits and at formal review intervals. In either case, there must be a close working association and joint reporting to both the project and the SMA organization.

NASA's Independent Verification and Validation (IV&V) is the third look at our most critical software. Engineering is responsible to build the software correctly and according to known good principles and thus is the first look. Software assurance works with the projects on a day to day bases, assessing the quality, safety, security and reliability of the processes and products and is the second look, with independent reporting chain up through the Center Safety and Mission Assurance Office and more closely associated with the total software processes and products. For NASA's most critical software, NASA's IV&V provides the third look, an objective examination of safety and mission critical software processes and products, delving into the analyses of the most critical aspects of the software on a project looking at safety, security and reliability. IV&V is considered to be technically, managerially and financially independent from the projects it works on. IV&V focuses on three perspectives:

- Will the system's software do what it is supposed to do?
- Will the system's software not do what it is not supposed to do?
- Will the system's software respond as expected under adverse conditions?

As a part of Software Assurance, IV&V plays a role in the overall NASA software risk mitigation strategy applied throughout the lifecycle, to improve the safety and quality of software systems.

Improvement of the software assurance program is achieved via four main paths and sundry smaller ways. First, there is a robust audit program that checks not only that the requirements are being followed in the field, but also brings the NASA Software Assurance Manager data to consider for systemic problems with the requirements implementation, training, and with the requirements themselves. Each of NASA's Centers and facilities, as well as some of the major programs are audited at least every 2-3 years from the Headquarters level. After each audit, the findings are discussed and compared to previous audits and center/facility results. Any repeat or evolving problem areas are then discussed with the Center/facility personnel for resolution and the data is used to see what additional training, guidance or even changes are needed in the requirements. At the Center or facility level, internal audits for each project are run more frequently according to the schedule and criticality of project development. In addition, Center Safety and Mission Assurance level technical authorities monitor all safety, reliability, quality, and software assurance on projects. Projects work with SMA to conduct internal audits and CMMI® Level 3 assessments (or equivalent) are required for all NASA's Class A, and most of Class B, Software projects.

The second improvement path is the NASA Software Assurance Research Program (SARP). Software engineering development and analyses continues to evolve, in order to stay current with software changes and the environment in which software is developed and operated, NASA has a long standing research program, SARP, which yearly polls the software assurance and software engineering communities for areas of need in software assurance. Then a research call is sent out, mainly within NASA, to solicit proposals to address these issues. SARP seeks practical solutions, tools, guidance, and processes of value to the greater software assurance community. The projects can be from 1 to 3 years in length with a transition to practice as part of the work. The proposals are peer reviewed by the community and selected according to need and meeting the SARP criteria for a good project. Some examples of SARP's output include a software (and system) hazard tracking system; guidance on better ways to collect, visualize, and present software assurance data; processes for performing Model Based testing of large systems; cost estimation methods for software assurance activities, and command reliability, to name a few. SARP usually has one or two projects that look to future software assurance needs, researching and posing potential solutions, and questions, to the ever evolving software development and operational landscape. Not every year are there sufficient funds to cast the research requests out to all industry and academia, but it is not insular, either. While limited, NASA's SARP program does seek out input from academia and beyond to keep current with new trends in software and computing systems.

Third, and most importantly, the NASA Software Assurance community is a close knit group that shares successes and failures, supporting one another and working together to create the assurance and safety standards, guides, and select needed training and research. Meeting, on average, twice a month via telecon and once a year in person, the NASA Software Assurance Manager, NASA Safety Center Software Assurance Lead and all the Center/facility software assurance leads and personnel stay current with Agency trends and needs. They review SARP work, present on their Center/Facility work, needs, and issues. Then they jointly formulate the NASA Software Assurance Objectives, Goals, Strategies and Metrics to create a road map to improve software assurance, laying out 1 to 5 year goals and strategies. This strong community is the heart of NASA software assurance.

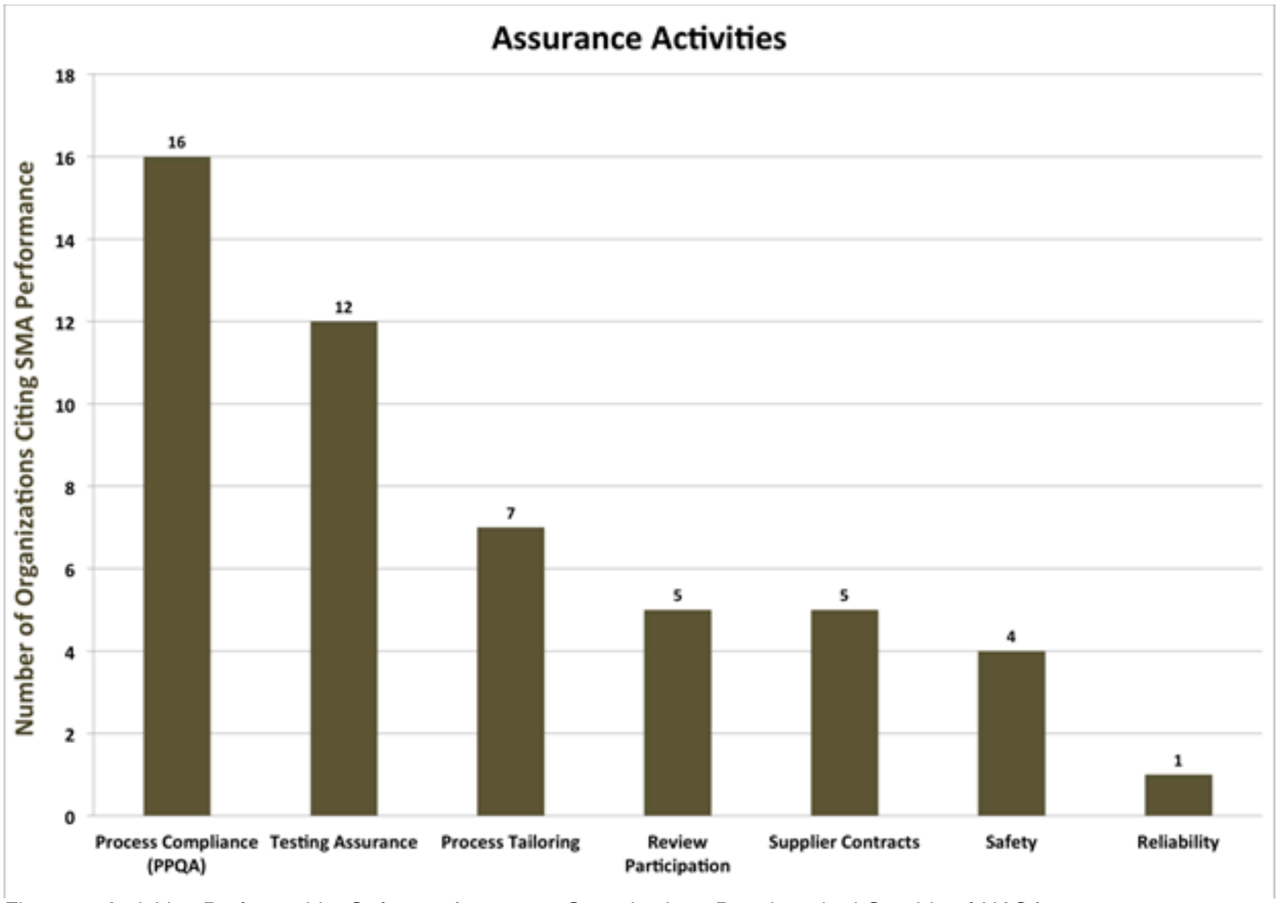


Figure 2: Activities Performed by Software Assurance Organizations Benchmarked Outside of NASA

The current NASA Software Assurance Objective: Demonstrate Software Assurance's contribution to assuring safety and mission success across all of NASA programs/projects/facilities with software.

Goal 1: Strengthen and maintain software assurance core competencies at all NASA centers [Make sure we have the right skills to do the job]

Goal 2: Establish a core set of SA performance measures for all Centers across the Agency [Measure if we are effective and if not, know where to fix it]

Goal 3: Increase value and establish SA as a core Engineering discipline [Provide clear relationship between what SA does and the risks it mitigates. Focus on risks the project and stakeholders need most to resolve.]

Goal 4: For all NASA Projects, obtain the appropriate level of SA funding necessary to meet the projects' software assurance requirements tailored to the Program/Project's risk posture [Assure sufficient software assurance on a project for the scope, criticality and classification of the software, which means knowing what it costs to perform SA.]

The NASA Software Assurance community, while close knit and supportive does look beyond the NASA problem field. Benchmarking of academia, industry and portions of DOD allows NASA to not only compare where we are with others, but infuse new ideas. SMA has benchmarked with the Navy in the past and learned and shared both problems and approaches to success. NASA software engineering and assurance have recently benchmarked with 18 organizations, five of them from industry. The main assurance activities reportedly performed by the benchmarked organizations can be seen in Figure 2, above. The numbers show how many of the 18 benchmarked organizations reported having software assurance involved with each activity listed. Note that two universities had no formal assurance role at all. Also note that the activities are not counted if performed by other roles. For example, engineers rather than assurance personnel are often assigned to software safety and reliability.

Although some activities may not be reflected in the numbers (topics missed in the discussion), they provide a starting point for examining software assurance as documented versus the actual practice of it. While NASA has much more focus on software safety and reliability within the assurance organization, as well as contributing to the acquisition process and verification and validation processes, those benchmarked against NASA usually saw those activities as falling within another group.

The five Aerospace Industry organizations interviews can be summarized as follows:

- All industry organizations reviewed saw the main function of software assurance as performing process and product quality assurance (PPQA) and maintaining software compliance with institutional standards. Safety and reliability were seen as engineering roles.
- These organizations also tended to have a low ratio of software assurance engineers to the number of developers. For example, one organization had five to six software assurance engineers for about 200 developers. At the far end, an organization had only one "SQA person" for a 100-person software engineering project
- The high CMMI process maturity of most of these organizations might be a factor in their perceived need for assurance. All but one had been appraised at CMMI Maturity Level 3 or higher and, as one noted, greater process maturity means more repeatable, institutionalized processes and fewer audit findings. The one organization that hadn't used the CMMI also used one assurance person for a team of 15 developers and 4 testers – the highest ratio in the group.
- The industry organizations tended to use tools and metrics on the engineering side. Two of the organizations mentioned wide use of Six-Sigma, which also correlates with high CMMI maturity.

Defense services organizations were interviewed, and their inputs on this topic can be summarized as follows:

- All the defense organizations had been appraised to some CMMI level; two had achieved CMMI ML 5 at some point, with one maintaining certification.
- Following a similar pattern to the industry organizations, all four organizations used software assurance primarily in a PPQA role and did not discuss their role in reliability or safety.
- Three out of the four organizations also used software assurance to witness or otherwise assure software testing.
- The one CMMI ML5 organization maintained a process assurance group, dedicated to process compliance, and QA and IV&V groups for checking products.
- Of the three organizations that discussed Field Programmable Gate Arrays (FPGAs) or other Programmable Logic Devices (PLDs), none mentioned software assurance.

The following trends were identified, based on these interviews and compared to 5 of the 10 NASA centers:

- NASA Centers tended to involve software assurance in a greater range of development activities. Four of the five assurance organizations were involved with process tailoring, in addition to the PPQA audits.
- Four out of the five (not the same four) were witnessing or otherwise assuring that tests were performed properly.
- Four out of five NASA Centers also performed some assurance activity related to software safety.
- Three out of five of the NASA Centers used assurance personnel to monitor suppliers or software contracts in some way.

From this particular benchmarking, NASA software assurance has a broader scope, even if some of the Centers are not as involved as others in all the software activities that NASA describes as software assurance. This can be explained in part by the allowance of software safety and reliability to be performed by other organizations but also, not all NASA Centers work on Software Class A or B software. Many software projects at the NASA research Centers are assuring research and technology development projects.

In today's environment of cyber attacks, NASA has, in the past, considered this to be the realm of the Chief Information Office and the Protective Services Office. This may have worked for us in the past, but in today's world, NASA's Software Assurance has a role to play as well. In the DoD world, the term "software assurance" has almost become synonymous with cyber security and their increased focus in this area is understandable as the effort is large and only getting bigger. We are all vulnerable, and for NASA, our software resources, especially software assurance, are limited. The NASA software community (engineering, assurance, project management) is now joining our CIO colleagues in reaching out to the forums, training and working groups of DHS, DoD, NIST, and others to accelerate our efforts and share what we have learned with those who are also in this struggle. Still, NASA, like our colleagues, must continue to provide the quality, reliability and safety aspects of software that has kept NASA flying for many years and which supports elimination of vulnerabilities. While NASA SA is working more closely with the CIO office to better cover security oversight of Mission Software, it has not given up its strong dedication to safety, reliability, quality and Independent Verification and Validation, rather it has incorporated mission software security assurance into its repertoire.

Disclaimer:

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

CIVILIAN TALENT IS MISSION-CRITICAL.
LET'S GET TO WORK.
Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.
You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.
Discover more about NAVAIR. Go to www.navair.navy.mil.
Equal Opportunity Employer | U.S. Citizenship Required
NAVAIR CIVILIAN
CHOICE IS YOURS.

ABOUT THE AUTHOR

Martha Wetherholt is the NASA Software Assurance Manager and Technical Fellow. She has worked at NASA since 1989 focusing her efforts in the areas of Software Safety, Reliability, Risk Management, and Quality. Prior to working at NASA, Martha worked predominately in the factory automation industry, working as a software engineer, systems engineer and supervisor, managing several software intense projects, balancing the need for delivering quality working products within schedule and resources. After many years in industry, Martha accepted the role of Lead in Software Product Assurance at NASA's Glenn Research Center. In 2001, she became the NASA Agency Software Assurance Manager at NASA Headquarters in Washington, DC. Her current roles and responsibilities include establishing the tasks, procedures, policies, training, and requirements for software assurance and safety across the Agency. She has led the effort to create and maintain the NASA SW Assurance and SW Safety Standards and Guidelines as well as establish a set of SW Assurance core competencies. Martha continues to work with NASA Center software assurance and safety personnel to strengthen the role of software assurance and software safety, assuring that the training, expectations, policies, guidelines and support is there to achieve this work. Martha has worked to bring software to its proper place in systems and facility safety. She established and maintains a core, Agency-wide SA team with whom she works to strengthen the role of SA, assuring that the support is there to meet NASA's software challenges. Martha has received many NASA awards including the prestigious NASA Medal for Exceptional Achievement for her work in software safety. She is a member of IEEE and AIAA. Martha received her BS degree in Biomedical Engineering from Case Western Reserve and a MS from Cleveland State in Industrial Engineering.

Software Security Assurance SOUP to NUTS

Dr. C. Warren Axelrod, Delta Risk LLC

Abstract. The ability to assess risks of and from specific software supply chains depends in large part on the amount, accuracy and availability of essential information. Only when such information is at hand can we hope to assure ourselves of the quality and security of installed software. In this paper we use an expanded version of the Cynefin Framework to come up with preferred approaches to categorizing software supply chains not only based on the potential knowledge levels of those responsible for evaluating, approving and operating systems, but also according to what can be known about particular supply chains. We suggest how each category of supply chain might be evaluated and fixed in the face of adverse incidents.

Introduction

For this context the most appropriate definition of "supply chain risk" is:

"... the risk that an adversary may sabotage, maliciously introduce unwanted function, or otherwise subvert the design, integrity, manufacturing, production, distribution, installation, operation, or maintenance of a covered system so as to surveil, deny, disrupt, or otherwise degrade the function, use, or operation of such system."[1]

In order to manage software supply-chain risk, accurate and extensive data must be collected, analyzed and responded to. All too often, however, crucial data are not readily at hand or they are difficult and/or expensive to collect, if indeed they can be gathered at all.

According to a 2004 report on "Defense Acquisitions,"[1] the GAO found that the U.S. DoD acquisition and software security policies were inadequate particularly in addressing risks relating to foreign suppliers developing weapon system software. Because of increasing difficulty and costs of testing computer code, the GAO suggested that, rather than testing code, those responsible for approving systems learn more about who developed the software and where they were located in order to arrive at a more informed vendor selection decision, which could mitigate risks. While such an approach is better than nothing, it does not come close to the level of software assurance obtained from independent in-depth testing of computer code. Furthermore, software makers usually incorporate software components from other sources, including open sources, which may not be known to vendors, contractors, or their customers.[2]

In this article, we investigate why so much necessary information is not forthcoming and propose approaches for obtaining elusive and costly software supply-chain data. Such information can provide analysts with the ability to anticipate, detect and react to adverse issues before, during and after they occur, rather than well after the fact, which is unfortunately more usually the case. Investment in the collection and analysis of software supply-chain metrics offers the potential of significant returns on investment in an ever more complex environment.

In addition, a worldwide data-sharing infrastructure is needed in order to allow entities comprising global supply chains to inform one another of events that will likely have a significant impact on the quality and availability of supplied software and equipment components. In order to understand what data need to be collected and how they should be used by decision-makers to manage the vagaries of software supply, we take the Cynefin Framework and extend it to cover additional software supply-chain characteristics. Based on this approach, we are able to suggest appropriate data-gathering and decision-making methods that meet each of a large variety of situations.

DoD and National Security Context

In a 2012 report on "IT Supply Chains," [3] the GAO affirmed that, among the four U.S. national security-related departments, the DoD had made greater progress by defining supply chain protection measures and implementing procedures for IT supply-chain assurance than had the departments of Energy, Homeland Security and Justice. Nevertheless there is still much work to be done by the latter three agencies with national-security responsibilities, as recommended by the GAO report.

This does not mean, however, that the DoD is free and clear when ii comes to IT supply-chain risk management. Despite all the progress in methods, procedures and tools that has been made over the last decade, there are still many areas that remain unknown, and may not even be knowable, to DoD program managers, particularly since extensive code reviews and software assurance testing have not been required. This implies that full assurance of IT supply chains remains a goal rather than a reality. Little has appeared in the literature on the ability of analysts to know each and every component of IT supply chains so that many of the structures of, and participants in, supply chains remain obscure or unknown, particularly with respect to commonalities [4]. Consequently, many vulnerabilities are not known either. As stated in [5]:

"[The DoD needs] to better "see" into some legs of the supply chain, especially where critical components are involved."

While a report by Adams [6] is oriented towards the manufacture of physical products rather than software in regard to supply chains of the U.S. defense industry, its conclusions also apply to IT products, software, and services. The report recommends the following:

1. Increase long-term federal investment in high-technology Industries
2. Apply and enforce existing laws and regulations
3. Develop domestic sources for key ... resources
4. Develop plans to strengthen the defense industrial base
5. Build consensus ... on the best ways to strengthen the defense industrial base
6. Increase cooperation among federal agencies and between government and industry
7. Strengthen collaboration among government, industry and academic research institutions
8. Ensure collaboration on economic and fiscal policies for long-term budgeting
9. Modernize and secure defense supply chains [emphasis added]
10. Identify potential defense supply-chain chokepoints and plan to prevent disruptions

It should be noted that the report [6] does not generally focus on the need to collect the knowledge necessary for making appropriate supply-chain decisions, although the exhortation to "identify chokepoints" implies some degree of information gathering. Facilitating the acquisition, analysis and understanding of data about software supply chains is a dominant objective of this article. That is to say, we want to bring to light how decision-makers should go about determining what is known, what is not known, what it will take to acquire the necessary knowledge, what is unknowable, and what they need to do under various circumstances.

Similarly, neither the recently published NIST Special Publication [7], which applies across all Federal information systems and organizations, nor the CNSS report [8], which addresses national security systems, specifically examine the "ability to know" supply-chain information. They proceed with the understanding that required information is readily available, which is far from the case in many circumstances. Nevertheless, both of these publications set forth invaluable guidance and the CNSS report [8] provides a very useful list of references with which DoD managers responsible for supply chains should become familiar.

The Provenance of Software

If you don't know where critical software comes from, then you may well be in the SOUP, literally, where SOUP means "Software of Unknown Provenance (or Pedigree)" Such software products may not be trustworthy because their origins are questionable or unknown. At the other extreme, if you think that you know everything about a particular piece of software, e.g., who designed it, who wrote it, and who tested it, the results of the tests, and so on, then you might be willing to rely on NUTS[3] or "Not Unreasonable Tracking Systems," in order to verify that the software development lifecycles (SDLCs) involved follow predetermined routes and are subject to appropriate levels of oversight.

Of course, there are many other situations between no knowledge and complete knowledge, such as knowing something about the backgrounds of some of the developers and their works, but not enough to give one much confidence that there aren't any little malware devils that might be lurking within the overall system, often for years, until they are revealed through some incident or other. Even when software is "open source," meaning that its source code is available to anyone wishing to look through the programs and modify them (under certain predetermined conditions), there are no guarantees that errors or deficiencies have not been introduced or that there is sufficient funding to provide suitable levels of technical and operational support. The exploitation of Heartbleed and Shellshock malware demonstrated this.[4]

Furthermore, there are times when everyone else appears to have known about some threat or vulnerability, but you just didn't happen to have been aware of them (oblivious), in which case there will some answering to do in order to satisfy management … or not, as the case may be.

Goals of Decision Makers

In order to establish the best possible situation, given the proliferation of buggy software and the ability of evildoers to take advantage of these deficiencies, one's goals should be to:

- determine what is known about a piece of software's provenance and what is not
- understand which risks are known to the community and which are not
- find out more about unfamiliar risks so that they might be mitigated
- take steps to mitigate known risks or have good reasons for not having done so
- come up with approaches for dealing with unknown or unexpected risks
- establish a professional and industry/sector network to stay informed about risks relating to supply chains of software that you plan to acquire and install
- maintain current knowledge about software supply-chain research, industry/sector and professional publications, conferences, podcasts, webinars, etc.
- understand that there are certain software products that operate covert systems about which you may never know but which can affect you in some way or another, purposely or inadvertently

We will gain a better understanding of how to achieve these goals by expanding an established decision framework to incorporate additional contexts found in software supply chains.

The Known/Unknown (K/U) Model

Since lack of knowledge is a major contributor to inadequate and inappropriate responses to supply-chain malfunctions and failures and the ability to recover quickly, it is important to fill in where there are clearly deficiencies. The first step is to understand what makes up the universe of knowledge and then determine which areas need to be augmented with a higher level of understanding. In Table 1, we show how knowledge about software supply chains might be categorized depending upon how knowledgeable cybersecurity professionals might be concerning particular software supply-chain deficiencies or weaknesses.

The underlying concept here is that either you know or don't know in advance about specific threats or vulnerabilities with respect to particular software products' supply chains. If you did know, the question then arises as to whether you responded appropriately. If you didn't know, then how are you going to ensure that you will get advance notification if and when a similar situation is occurs in the future? If you didn't know but should have known, then your suitability to the task is in question. If you could not have known, you need to examine whether you have appropriate monitoring and incident-response mechanisms in place to react correctly.

These concepts of whether one is aware or unaware of various situations have been incorporated into a framework, called the Cynefin Knowledge Framework ("Cynefin"), which is designed to assist leaders in their decision making. The model is described in [9]. As mentioned above, we will expand this framework to facilitate decision-making with respect to software supply chains.

The Cynefin Knowledge Framework

Cynefin (translated from the Welsh as "habitat" or "place") is roughly analogous to the above K/U model. Cynefin suggests how decision-makers should respond to events that fall within

Analysts' Knowledge	Information Available	
	Knowns	**Unknowns**
Known	**Obvious** – I knew all about this in advance but didn't act on it quickly enough	**Obscure** – I knew that I didn't know anything about this, but couldn't get the data for economic or other reasons
Unknown	**Oblivious** – I was not aware of this even though my peers were	**Unfathomable** – I didn't have a clue that this existed, nor did my peers

Table 1. K/U model categories of knowledge by information available

various contexts. In this article, we extend the framework to cover situations not specifically addressed in Cynefin.

In a video,[5] Snowden differentiates between categorization models (such as the 2 x 2 matrix K/U model above) and "sense-making" frameworks, such as Cynefin. With categorization models, the framework precedes the data; but for sense-making frameworks, "the framework itself emerges from the data ..." Figure 1 illustrates Cynefin, which has evolved over time.[6] For example, "simple" contexts in have been replaced with "obvious" contexts, and the fifth category "disorder" seems to have been dropped. Also, there are subtleties that do not show up in the diagram, but are described in the video, such as catastrophic consequences of a transition from "obvious" to "chaotic" contexts. "Disorder" contexts cover otherwise uncategorized items.

Cynefin divides the contexts between "ordered systems," which are highly constrained and predictable, whether obvious or complicated contexts, and "unordered systems," which have fewer constraints and, for chaotic contexts, exhibit unpredictable random behavior.

Categories within the known/unknown (K/U) model are quite similar to Cynefin contexts, except for two instances. One instance is the chaotic system, the context of which is "unknowable," and the other instance is K/U Model's category of "unknown knowns," which is not represented explicitly in Cynefin. Table 2 shows similarities and differences between the two models.

In Table 2, we have added three contexts, namely, "oblivious," "obscure," and "stealth," which are shown in the shaded entries. "Oblivious" contexts, which are part of the K/U model, are those where decision-makers are not aware of certain information generally known to many practitioners. Note that "oblivious" is a characteristic of decision-makers rather than of systems. "Obscure" contexts, which belong to neither Cynefin nor K/U, are those where surreptitious methods are needed to find out about system vulnerabilities.[7] "Stealth" contexts are for systems which are meant to be kept secret.[8] The expanded Cynefin framework is illustrated in Figure 2.

According to the definitions of Cynefin realms, "knowable "and "known unknowns" realms are equivalent—for "knowable," decision-makers are aware that certain items, which are not known, may become known through analysis. For "known unknowns," items are known to some but not to others.

If items are "unknowable," then nobody knows about them and you are generally "off the hook" if they occur. However, if you

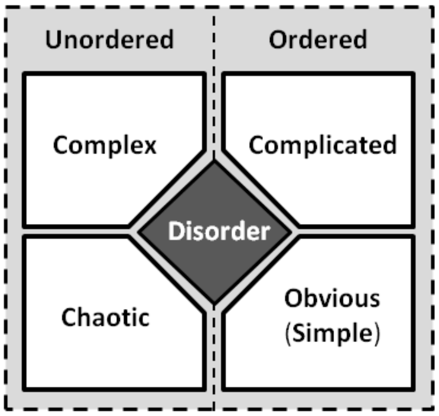


Figure 1: The Cynefin Knowledge Framework

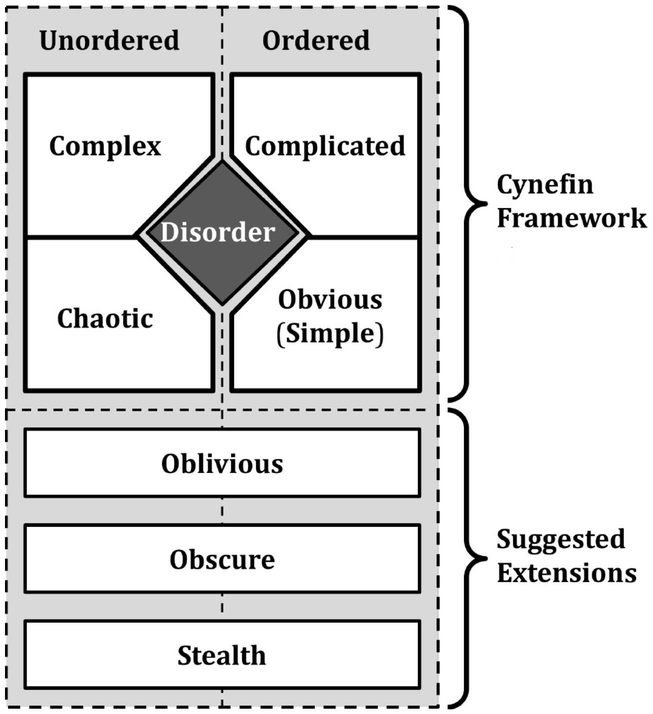


Figure 2. Suggested Expanded Framework

Contexts	Practices	K/U Model	Cynefin Realms	Differentiating Activities
Obvious	Best	Known knowns	Known knowns	Categorize
Complicated	Good	Known unknowns	Knowables – Known unknowns	Analyze
Complex	Emergent	Unknown unknowns	Unknown unknowns	Probe
Chaotic	Novel	--	Unknowable unknowns	Act
Oblivious	Ignorant	Unknown knowns	--	Investigate
Obscure	Clandestine	Unknown knowables	--	Deal
Stealth	Secret	Unknowable unknowables	--	Respond

Table 2: Knowledge for system contexts of an extended framework.

don't know about something that you should (this is not addressed specifically in Cynefin), then you might be accused of not maintaining currency in the field. This latter situation is most dangerous with respect to software supply chains, since decision-makers might be considered ignorant (or worse, negligent) in the event that something goes wrong.[9] This is why information sharing is crucial for successful management of software supply chains.

We now extend Cynefin to include the K/U model so as to determine the decisions that need to be made and the amount of effort to be expended on assessing and mitigating risks. In Table 3, we show Cynefin (unshaded areas) with extensions derived from the K/U model (shaded areas).

Realistically, there are those with software supply-chain responsibilities who are somewhat unaware of what is going on in the outside world as it pertains to their supply chains. Published reports about how organizations scramble in response to malware and hacking incidents and other forms of supply-chain disruption support the contention of ignorance, even when information about vulnerabilities and weaknesses are already in the public domain.[10]

Most academic treatments of this topic do not address dealing with criminal elements to obtain obscure information about malware and back doors that may have been inserted into software products during their supply-chain lifecycles. However, it is common knowledge that there are large and lucrative markets for the sale of exploits and vulnerability information.[11] Some might consider such information to be "unknowable," if they refuse to deal with dubious, clandestine or criminal elements. Also, the news about secret software systems is usually mere happenstance as might occur through some error or by the leakage of classified information by insiders.

Software Supply-Chain Risks

It can be difficult to come up with meaningful risk assessments for each of the seven contexts in the extended Cynefin. In the first place, analysts and/or decision-makers are often not aware of supply-chain weaknesses. Whether such defects will have serious personal and organizational consequences depends largely on efforts made to find out about vulnerabilities preemptively. As mentioned, an important consideration is whether one's peer group is already aware of such vulnerabilities. It is much more damaging to one's career if you are one of only a very few who lack knowledge than in a situation where everyone is just as ignorant.

The reverse may not be true, however. If you anticipate an issue that others don't or won't recognize as important, whether it is to your advantage or not when an incident occurs depends on whether you acted on the knowledge. For example, if you expect the electrical power grid to be unreliable in a particular country or region and you installed a generator when others in the area did not, you become a hero when a power outage occurs. However, if you just mentioned the power problem but did not install a backup generator, you might be considerably worse off than if you hadn't mentioned the problem in the first place, since you might be accused of not being aggressive enough in making your case.

Software Supply-Chain Lifecycles

As described in [12], software supply chains differ significantly from those of physical products. Software's unique characteristics include the following:

- Software can be copied without affecting the original and sold on the black market
- Software can be distributed in electronic form without transporting physical media
- Malware and back doors can be inserted into authentic software without leaving any trace

Because of these characteristics, the software supply-chain lifecycle is also somewhat unique. Table 4 lists specific attributes of software supply chains for each phase.

Information and Communications Technology (ICT) Supply-Chain Risks

A particularly extensive report [13], developed by the DoD, provides a list of threats that can, and do, impact software and software supply chains, including: Sabotage, Tampering, Counterfeiting, Piracy, Theft, Destruction, Disruption, Exfiltration—theft, Exfiltration—disruption, Infiltration, Subversion, Diversion, Export Control Violations, Corruption, Social Engineering, Insider Threat, Pseudo-insider Threat, and Foreign Ownership.

Ordered/ Unordered	Knowledge	Knowns	Unknowns
Ordered	Known (Knowable)	• Contexts: **Obvious** (Simple) • Realm: **Known knowns** • Domain: **Best practice** • **Standard process** invoked with review cycle & clear measures	• Contexts: **Complicated** • Realm: **Known unknowns** • Domain: **Good practice** • **Analytical techniques** used to determine facts
Ordered or Unordered	Unknown	• Contexts: **Oblivious** • Realm: **Unknown knowns** • Domain: **Ignorant** • **Investigations** of vendors, contractors and industry and professional groups to find out what is generally known	• Contexts: **Complex** • Realm: **Unknown unknowns** • Domain: **Emergent** • **Diverse interventions** needed to create options
Ordered or Unordered	Unknowable	• Contexts: **Obscure** • Realm: **Unknowable knowns** • Domain **Clandestine** • **Clandestine** dealings to try to get information	• Contexts: **Chaotic** • Realm: **Unknowable unknowns** • Domain: **Novel** • **Single or multiple actions** required to stabilize situation
Ordered or Unordered	Unknowable		• Contexts: **Stealth** • Realm: **Unknowable unknowables** • Domain: **Secret** • Able to **respond** only when secret is unintentionally disclosed

Table 3: Extensiaons to the Cynefin framework compared to the K/U model

Phase	Software Supply-Chain Lifecycle Attributes
Requirements	Requirements (specifications), design and development can be done virtually anywhere that has suitably educated staff and reliable, low cost telecommunications
Design	
Building (Development)	
Distribution	Although some software is still distributed on physical media, it is common to distribute software electronically and increasingly software is available in the Cloud so no distribution as such is necessary.
Warehousing	
Deployment	Software is deployed via various wholesale and retail outlets although it is often downloaded from vendor and or distributor websites, including open-source.
Operation	In theory, software can be run indefinitely although there are reasons for it becoming obsolete, such as cessation of vendor support, replacement of operating systems and platforms, changes in hardware, etc.
Maintenance and Support	
Disposal	Software can generally be deleted or replaced without having to destroy media, although having users properly eliminate all traces of the software, including backup copies, is unreliable.

Table 4: Software characteristics for phases of the supply-chain lifecycle

While many of these threats apply to software products generally, including those built in-house, they all can occur in both national and global software supply chains. Table 5 suggests some risk mitigation approaches for each context of our extended model:

In general, risk mitigation comprises obtaining as much advance warning as possible from a broad population of sources and responding in ways that improve, rather than exacerbate the situation. It is strongly advised to have a complete set of contingency plans in place so that they can be drawn upon as circumstances require.

Software Assurance Factors

Much of software supply-chain risk management involves information sharing and decision making based upon contexts in order to mitigate the many risks that affect software supply chains. However, many incidents that occur can be avoided by proactively making sure that the software goes through a rigorous software assurance process, which might include various forms of certification.

Knowledge	Knowns	Unknowns
Known **(Knowable)**	**Obvious**—Activate preplanned response procedures which should have been developed as part of the software acquisition process	**Complicated**— • Try to avoid using particular software that is known to have issues (although specific issues may not be know) • If use is unavoidable, monitor status of software and apply patches immediately
Unknown	**Oblivious**—Activate incident-response procedures and quickly link up with professional and industry "grapevines" so as to be forewarned of future threats	**Complex**—Activate incident-response process and try to determine whether similar incidents might be anticipated and avoided in the future
Unknowable	**Clandestine**—Determine who might know about unknowable vulnerabilities and make deals with those with relevant information	**Chaotic**—React to unexpected chaos with creative responses in order to stabilize the situation before being able to take corrective or restorative actions
Unknowable		**Secret**—First, understand the relevance of the revelation of a secret system to your organization and then respond as appropriate, if at all

Table 5: Risk mitigation approaches for various contexts

In order to incorporate software assurance standards into supply chains, it is first necessary to determine what those standards should be and how they should be used and managed. As described in [10], this could be accomplished addressing a number of technical, economic and governance issues including:

- Development of software assurance technical standards
- Management of software-assurance and certification standards
- Evaluation of tools and techniques for assuring software
- Determination of update frequency for tools and techniques
- Focus on the most pressing threats to software and supply chains
- Establishment of models of the economics of software-assurance solutions, and testing and certifying software

Once such standards have been established, we come to the far greater task of enforcing them on third parties both domestically and internationally. As can be imagined, this would require a major political effort far beyond anything that has been attempted so far in this arena. Nevertheless, some significant part of this goal needs to be implemented if trust in software is to be achieved at even a rudimentary level. The only real possibility to make progress here is to use economic means of encouragement as can be brought about with a carrot, by (for example) requiring government agencies only to buy software that meets agreed-upon international standards, or with a stick by invoking legal measures that places liability on software manufacturers, as suggested in [10].

Conclusions

Before one can reasonably address the quality of software emanating from supply chains, it is necessary to understand the various contexts within which knowledge of software products' provenance can exist. It is suggested that the known/known model combined with the Cynefin framework can provide a basis for decision-making possibilities.

Risks relating to software supply chains come from both the software itself and the supply-chain process that served to create the software. We looked at many of these risks and suggested how they might be addressed.

Finally we looked at software assurance requirements that, if addressed appropriately into software supply chains, would serve to ensure that the software products themselves have the desired security and integrity.

In general, we are far behind where we should be in the fight against vulnerable and dangerous software and the practices that govern them. We therefore need to take a holistic view of the factors that affect software supply chains and the software products that emanate from them, and we must mitigate the risks with due deference to the need for efficient and effective means of manufacturing the software that is at the base of practically all new systems of any importance.

ABOUT THE AUTHOR

Dr. C. Warren Axelrod is a senior consultant with Delta Risk LLC specializing in cyber security, risk management and business resiliency. Previously, he was the Business Information Security Officer and Chief Privacy Officer for US Trust.

He was a founding member of the FS/ISAC (Financial Services Information Sharing and Analysis Center) and represented national financial services cyber security interests during the Y2K date rollover. He testified before Congress in 2001 on cyber security.

His recent books include Engineering Safe and Secure Software Systems (Artech House, 2012) and Outsourcing Information Security (Artech House, 2004).

He holds a Ph.D. in managerial economics from Cornell University, and a B.Sc. in electrical engineering and an M.A. in economics and statistics from Glasgow University. He is certified as a CISSP and CISM.

Phone: 917-670-1720

Email: waxelrod@delta-risk.net

ADDITIONAL READING

1. Axelrod, C. Warren. "Malware, 'Weakware,' and the Security of Software Supply Chains," CrossTalk (March/April 2014): 20-24.
2. Axelrod, C. Warren. "Addressing Supply-Chain Complexity Using Closed-Loop Simulation-Based Exercises," Proc, of the Complex Systems 2015 Conference, New Forest, UK, May 2015.
3. Davidson, Don, "Managing Global Supply Chain Risk: Security & Resiliency (of the Chain) and Integrity (of Product)," <http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Past%20Meetings/Program%20Protection%20Workshop%20May%201-2,%202012/Don_Davidson_ManagingGlobalSupplyChainRisk.pdf>

REFERENCES

1. U.S. General Accounting Office (GAO), Defense Acquisitions: Knowledge of Software Suppliers Needed to Manage Risks, GAO-04-678, May 2004.
2. Vaughan, Steven J. "It's an Open Source World: 78 Percent of Companies Run Open-Source Software," April 6, 2015. Available at <http://www.zdnet.com/article/its-an-open-source-world-78-percent-of-companies-run-open-source-software/>
3. U.S. General Accountability Office (GAO), IT Supply Chain: National Security-Related, Agencies Need to Better Address Risks, GAO-12-361, March 2012.
4. Axelrod, C. Warren. "Risks of Unrecognized Commonalities in the Information Technology Supply Chain," Proceedings of the 2010 IEEE International Conference – Technologies for Homeland Security, Waltham, MA, November 2010.
5. Davidson, Don, and Stephanie Shankles. "We Cannot Blindly Reap the Benefits of a Globalized ICT Supply Chain," CrossTalk, (March/April 2013): 4-7.
6. Adams, John, ReMaking American Security: Supply Chain Vulnerabilities & National Security Risks Across the U.S. Defense Industrial Base, Alliance for American Manufacturing, May 2013.
7. Boyens, Jon et al. Supply Chain Risk Management Practices for Federal Information Systems and Organizations, NIST Special Publication 800-161, U.S. Department of Commerce, April 2015.
8. Committee on National Security Systems (CNSS), Supply Chain Risk Management (SCRM), CNSSD No. 505, March 2012.
9. Snowden, David J., and Mary E. Boone. "A Leader's Framework for Decision Making," Harvard Business Review (November 2007).
10. Denning, Dorothy E. "Privacy and Security: Toward More Secure Software," Communications of the ACM, April 2015, pages 24-26.
11. Verizon Enterprise Solutions, 2015 Data Breach Investigations Report, 2015. Available via link at <http://www.verizonenterprise.com/DBIR/2015/>
12. Axelrod, C. Warren. "Mitigating Software Supply Chain Risk," ISACA JOnline, August, 2013. Available at <http://www.isaca.org/Journal/archives/2013/Volume-4/Pages/JOnline-Mitigating-Software-Supply-Chain-Risk.aspx>
13. Goertzel, Karen M., et al. State of the Art Report on Supply Chain Risk Management for the Off-the-Shelf (OTS) Information and Communications Technology (ICT) Supply Chain, Department of Defense, Information Assurance Technology Analysis Center (IATAC), USA, 2010.
14. Axelrod, C. Warren. "Reducing Software Assurance Risks for Security-Critical and Safely-Critical Systems," Proc of the 2014 IEEE LISAT (Long Island Systems, Applications and Technology) Conference, Farmingdale, NY, May 2014.

NOTES

1. This definition is from Section 806 of the Ike Skelton National Defense Authorization Act for Fiscal Year 2011 which is available at <http://www.gpo.gov/fdsys/pkg/BILLS-111hr6523enr/pdf/BILLS-111hr6523enr.pdf>
2. It is claimed in Vaughan [1] that 78 percent of companies use open-source components. While the source code of open-source software is readily available for viewing and testing, there remain many unknown issues with respect to the provenance, quality and support of particular widely-used software as recent incidents have shown.
3. The acronym NUTS already exists with several connotations, one of which is used by the military and has the meaning "Nuclear Utilization Target Selection." The author's designation of NUTS as "Not Unreasonable Tracking Systems" does not have any such provenance.
4. Two of the most damaging cyber attacks in recent times occurred against open-source software, namely OpenSSL (Heartbleed) and Bash (Shellshock). OpenSSL runs on a substantial population of web servers and Bash is integrated into many popular operating systems.
5. David Snowden's explanation of his framework is at <https://www.youtube.com/watch?v=N7oz366X0-8>
6. See Craig Brougham's July 9, 2014 posting "Cynefin 101–An Introduction," available at <http://www.infoq.com/articles/cynefin-introduction>
7. In [10], Denning mentions having the U.S. government pay "bug bounties" to obtain information about software vulnerabilities that they would then make available to the public, but she opposes such a program in favor of developing a suitable liability regime for software developers and users.
8. Stuxnet is an example of covert malware which was supposed to be kept secret but was accidentally released into the general Internet and was then analyzed and publicized, thereby losing much of its value by alerting potential victims as to its form and function.
9. There are many situations in which culpability depends upon who knew what and upon what one might reasonably be expected to have known at the time of an incident. The knowledge gap is attributable to decision-makers in such cases and not to contexts. This is perhaps why Cynefin does not include the "unknown known" category.
10. Verizon's latest data breach report [11] indicates that "99.9% of the exploited vulnerabilities were compromised more than a year after the CVE (Common Vulnerabilities and Exposure) was published."
11. See [10]

They Know Your Weaknesses – Do You?: Reintroducing Common Weakness Enumeration

Yan Wu, Bowling Green State University
Irena Bojanova, University of Maryland, Baltimore County
Yaacov Yesha, University of Maryland University College

Abstract: Knowing what makes your software systems vulnerable to attacks is critical, as software vulnerabilities hurt security, reliability, and availability of the system as a whole. The Common Weakness Enumeration (CWE), a community effort that provides the foundation for such knowledge, is not sufficient, accurate and precise enough to serve as the common language measuring stick and provide a common baseline for developers and security practitioners. In this article, we introduce the relevant body of knowledge that consolidates CWE, including the Semantic Template and Software Fault Pattern efforts, and how static analysis tools add value through CWEs. We also provide future directions, present our vision on CWE formalization, and discuss the value of CWE for not only software assurance community, but also for Computer Science.

1. Introduction to Common Weakness Enumeration (CWE)

Software weaknesses could be exploited to compromise a system's security. This is especially critical for systems such as the Department of Defense (DoD) systems, in which the amount of software is very large. Software assurance countermeasures should be applied to address anticipated attacks against a system. Such attacks are enabled by software vulnerabilities, and those countermeasures reduce those vulnerabilities or remove them[12].

Common Weakness Enumeration (CWE) [1] is a collection of software weakness descriptions that offers a way to identify and eliminate vulnerabilities in computer systems. CWE is also used to evaluate the tools and services developed for finding weaknesses in software. CWE is community-developed and maintained by MITRE Corporation [1].

A preliminary classification of vulnerabilities, attacks, and related concepts was developed by MITRE's CVE [2] team. That effort began in 2005., CWE was developed as a list of software weaknesses that is more suitable for software security assessment [14].

1.1 History of CWE

There have been several community efforts to leverage the existing large number of diverse real-world vulnerabilities. For example, an important step towards creating the needed collection of software weakness types was the establishment of the CVE (Common Vulnerabilities and Exposures) list [2] in 1999 by MITRE. Another important step from MITRE was creating the Preliminary List Of Vulnerability Examples for Researchers (PLOVER) in 2005. PLOVER includes more than 1,500 CVE names, and 290 types of software weaknesses. The organization of those vulnerabilities is based on the types of weaknesses among 290 types that cause each vulnerability [1].

The consolidation and evolution process of CWE [1] occurred during earlier efforts to classify vulnerabilities by answering three basic questions:

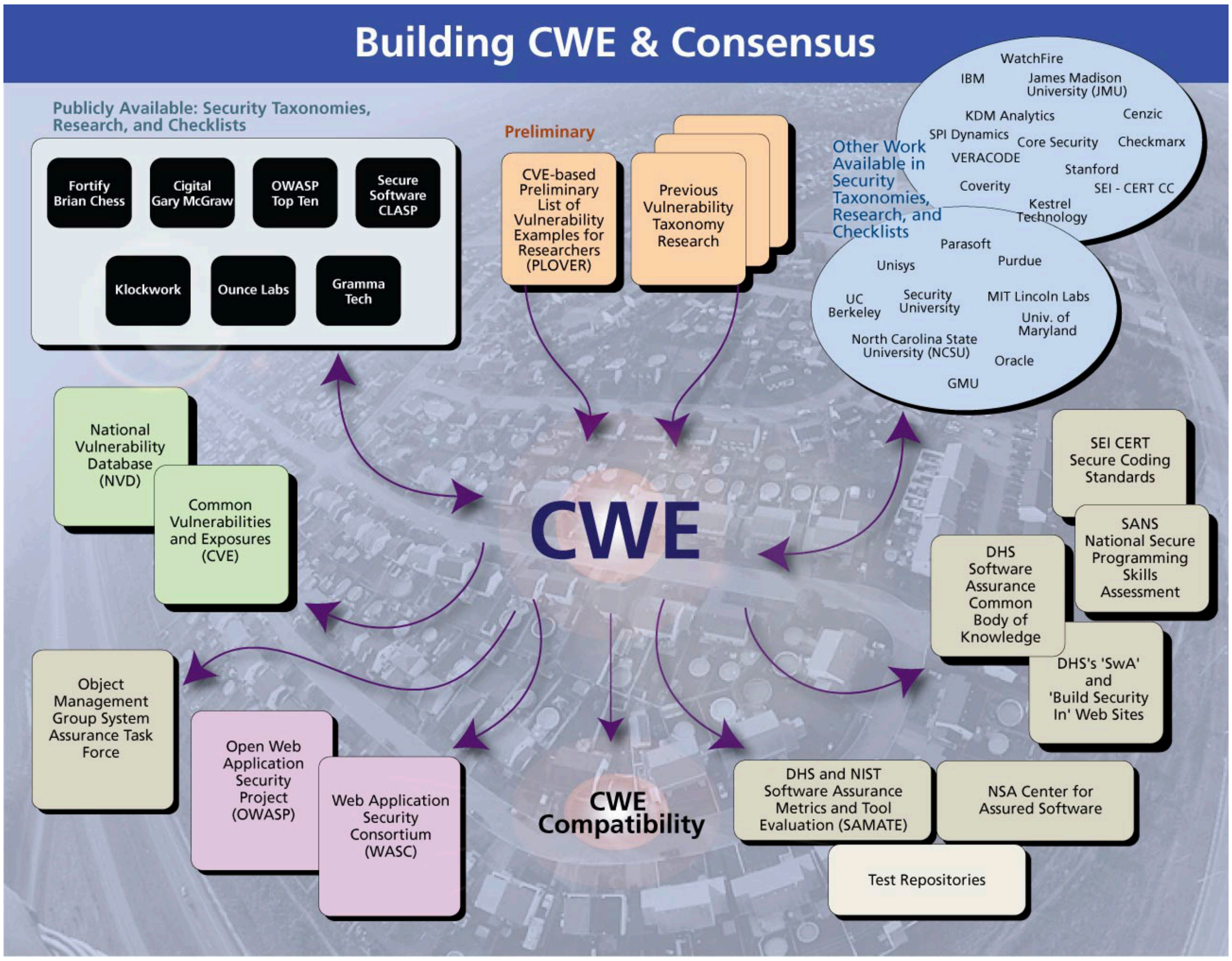
1) How did the vulnerability enter the system?
2) When did the vulnerability enter the system?
3) Where does the vulnerability appear? Or - Where is the vulnerability now?

Over a period of time, other revisions and ways to classify vulnerabilities were introduced. Until more recently, vulnerability categorizations have been developed as enumerations of weaknesses.

The CWE vision is to consolidate these efforts, and it is often compared to a "Kitchen Sink", although in a good way, as it aggregates many different taxonomies, software technologies and products, and categorization perspectives. While it provides a comprehensive record of software weaknesses, it can be a daunting task for developers to untangle the complex web of interdependencies that exist among software weaknesses captured in the CWE.

Figure 1 presents the CWE efforts context and community.

Figure 1. CWE Efforts Context and Community [http://cwe.mitre.org [1]]


Building CWE & Consensus
Publicly Available: Security Taxonomies, Research, and Checklists
Fortify Brian Chess
Cigital Gary McGraw
OWASP Top Ten
Secure Software CLASP
Klockwork
Ounce Labs
Gramma Tech
Preliminary
CVE-based Preliminary List of Vulnerability Examples for Researchers (PLOVER)
Previous Vulnerability Taxonomy Research
Other Work Available in Security Taxonomies, Research, and Checklists
WatchFire
IBM
James Madison University (JMU)
KDM Analytics
Cenzic
SPI Dynamics
Core Security
Checkmarx
VERACODE
Stanford
Coverity
SEI - CERT CC
Kestrel Technology
Parasoft
Purdue
Unisys
UC Berkeley
Security University
MIT Lincoln Labs
Univ. of Maryland
North Carolina State University (NCSU)
Oracle
GMU
National Vulnerability Database (NVD)
Common Vulnerabilities and Exposures (CVE)
CWE
SEI CERT Secure Coding Standards
SANS National Secure Programming Skills Assessment
DHS Software Assurance Common Body of Knowledge
DHS's 'SwA' and 'Build Security In' Web Sites
Object Management Group System Assurance Task Force
Open Web Application Security Project (OWASP)
Web Application Security Consortium (WASC)
CWE Compatibility
DHS and NIST Software Assurance Metrics and Tool Evaluation (SAMATE)
NSA Center for Assured Software
Test Repositories

1.2 CWE Concepts

Common Weakness Enumeration (CWE) [1] is a collection of descriptions of software weakness types stored as .xml, .xsd and .pdf documents. There are four major types of CWE-IDs: 1) Category, 2) Compound Element, 3) View, and 4) Weakness. The weaknesses covered by CWE have weakness IDs. Category and Compound Element are aggregations of weaknesses. Category aggregates types of weaknesses, and Compound Element aggregates a group of several events that together can result in a successful attack. View IDs are "assigned to predefined perspectives with which one might look at the weaknesses in CWE." [1]

Information provided for CWEs includes:

- CWE Identifier Number/Name of the weakness type
- Description of the type
- Alternate terms for the weakness
- Description of the behavior of the weakness
- Description of the exploit of the weakness
- Likelihood of exploit for the weakness
- Description of the consequences of the exploit
- Potential mitigations
- Node relationship information
- Source taxonomies
- Code samples for the languages/architectures
- CVE Identifier numbers of vulnerabilities for which that type of weakness exists
- References [1].

2. CWE Related Practices

Around CWE, there is a list of relevant body of knowledge such as Common Weakness Scoring System (CWSS), Common Vulnerabilities and Exposures (CVE), and Common Attack Pattern Enumeration and Classification (CAPEC). They are utilized by many institutions, including DoD, to identify and mitigate the most dangerous types of vulnerabilities in the software [12]

2.1 Use of CWE

CWE was established for those who create software, analyze software for security flaws, and provide tools and services for finding and defending against security flaws in software [1]. The CWE Compatibility and Effectiveness Program is based on six requirements: 1) "CWE Searchable," 2) "CWE Output," 3) "Mapping Accuracy," 4) "CWE Documentation," 5) "CWE Coverage," and 6) "CWE Test Results."

Meeting the first four requirements is needed for a product or a service to be designated as "CWE Compatible," and meeting all six requirements is needed for a product or service to be designated as "CWE Effective." [1] Static analysis tools are also encouraged to map their reports to corresponding CWEs so that the results from different tools could have a standard baseline to be matched and compared.

2.2 Common Weakness Scoring System (CWSS)

The Common Weakness Scoring System (CWSS) [3] is included in CWE project. Numerically scoring software weaknesses is important, as both software developers and software consumers need to compare weaknesses in order to prioritize among various activities related to avoiding and eliminating them. CWSS enables such scoring by methods such as: Targeted, Generalized, Context-adjusted, and aggregated. CWSS 0.8 is based on the Targeted scoring method. This method is applicable to a particular package. The CWSS 0.8 scoring formula includes eighteen factors, which are divided into three groups: The Base Finding Group, the Attack Surface Group, and the Environmental Group.

2.3 Common Vulnerabilities and Exposures (CVE)

CVE is a dictionary of security vulnerabilities. It was established in 1999 in response to lack of standardization of names of vulnerabilities: different repositories could refer to the same vulnerability by a different name, resulting in difficulty in comparing software security tools.

CVE provides standard identifiers for security vulnerabilities [2], and help in finding information about a vulnerability, including ways of, and available products for, eliminating the vulnerability. It can also help in determining whether particular tools are adequate for detecting attacks that are based on particular vulnerabilities [2].

After discovering a potential security vulnerability, a CVE Numbering Authority (CNA) can assign to it a CVE identifier [2]. Then the CVE Editor posts the information on the CVE List. The Primary CNA is MITRE Corporation. Other CNAs are software vendors, (for example, Apple Inc. and Adobe Systems Incorporated), third-party coordinators, (for example, CERT/CC), or researchers (for example, Core Security Technologies). The CVE Editor is MITRE Corporation.

2.4 Common Attack Pattern Enumeration and Classification (CAPEC)

Common Attack Pattern Enumeration and Classification (CAPEC) [4] was released in 2007. It includes descriptions of attack patterns. Information provided by CAPEC is needed in the process of finding vulnerabilities in software. In order to protect against attacks, knowledge of attack patterns is valuable, in addition to knowledge of software weaknesses that can be exploited by such attacks.

3. CWE in Practice

This section describes how the static analysis tools use CWEs to tag their tool reports and why it can add value to their products.

CWE contains a fairly comprehensive collection of application architecture, design, code, and deployment errors along with mitigation advice and examples of vulnerable and correct code segments. It also describes the SANS top 25 most dangerous software errors, that often "allow attackers to completely take over the software, steal data, or prevent the software from working at all." [1]

Because of its usefulness, CWE is already recognized and adopted by many organizations. For example, 40 organizations with 71 products and services already participated in the CWE Compatibility and Effectiveness Program (http://cwe.mitre.org/compatible/organizations.html). CWE has been adopted by NIST's National Vulnerability Database (NVD) (http://nvd.nist.gov) with mappings between CVEs and CWEs, and the Open Web Application Security Project (OWASP) – Top Ten Project (https://www.owasp.org/index.php/owasp_top_ten_project). Also, as part of the NIST SAMATE project, warnings from different tools that refer to the same weakness are being matched to corresponding CWE IDs to facilitate tools evaluation [9].

State-of-the-art static analysis tools today are able to find significant types of software security weaknesses. Many tools that support CWE are accompanied by public listings of the CWEs, and they are effective at finding and tag their vulnerability reports with corresponding CWE IDs. However, some mappings are not very precise, as CWE is organized into a hierarchy and some weakness types are refinements of other weakness types; also a single vulnerability may be the result of a chain of weaknesses or the composite effect of several weaknesses. The reality is that no single tool can detect all weaknesses and multiple tools should be used for complete coverage and better they all support CWE identification to facilitate the communication among them.

Customers also ask for the mappings of found weaknesses to the CWE IDs, as this provides common grounds for evaluating tools' performance and weaknesses' coverage. Therefore, even Static Analysis Tools that claim to be responsible for only limited number of weakness types [1] should not underestimate the importance of CWE and the mappings to CWE IDs.

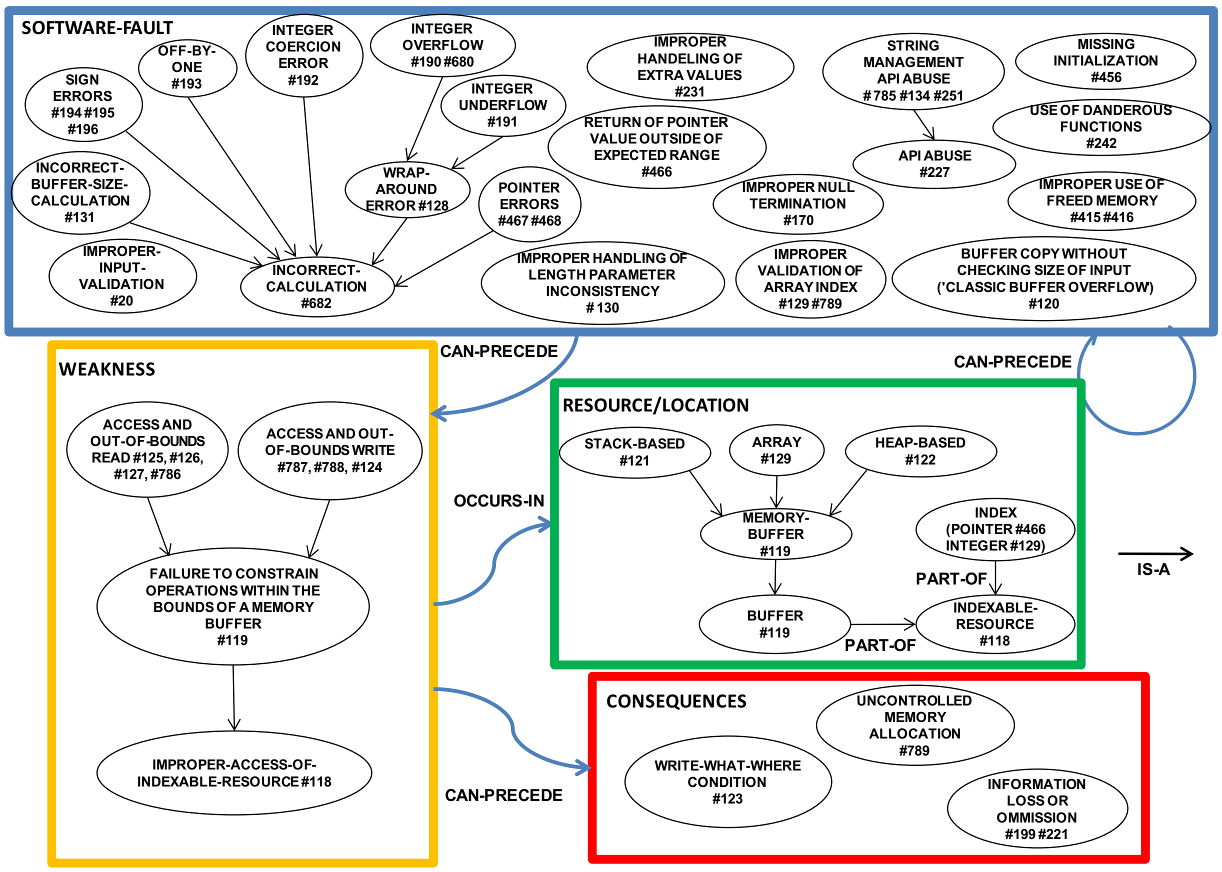
4. Improving CWE

This section describes existing efforts, which include Semantic Template and Software Fault Pattern, to improve the readability and usability of CWEs.

CWE is a collection of weaknesses with a highly tangled structure at various levels of abstraction, mixed contents of attack, behavior, feature, flaws, and all by natural language representations. It means that using its relatively unstructured weakness categories is a daunting task for stakeholders in the software development community. To help utilize the valuable contents of CWE, efforts have been made by both academia and industry to improve the readability and usability of the CWE.

Wu et. al. [5] reorganized categories of CWEs into Semantic Templates to help developers and researchers construct a more clear mental model and improve the understanding of weaknesses. To facilitate the CWE use in the study of vulnerabilities, easy-to-understand templates for each conceptually

Figure 2. Buffer Overflow Semantic Template


SOFTWARE-FAULT
SIGN ERRORS #194 #195 #196
OFF-BY-ONE #193
INTEGER COERCION ERROR #192
INTEGER OVERFLOW #190 #680
INTEGER UNDERFLOW #191
IMPROPER HANDELING OF EXTRA VALUES #231
STRING MANAGEMENT API ABUSE #785 #134 #251
MISSING INITIALIZATION #456
INCORRECT-BUFFER-SIZE-CALCULATION #131
WRAP-AROUND ERROR #128
POINTER ERRORS #467 #468
RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE #466
IMPROPER NULL TERMINATION #170
API ABUSE #227
USE OF DANDEROUS FUNCTIONS #242
IMPROPER USE OF FREED MEMORY #415 #416
IMPROPER-INPUT-VALIDATION #20
INCORRECT-CALCULATION #682
IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY # 130
IMPROPER VALIDATION OF ARRAY INDEX #129 #789
BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW) #120
CAN-PRECEDE
CAN-PRECEDE
WEAKNESS
ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786
ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124
FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119
IMPROPER-ACCESS-OF-INDEXABLE-RESOURCE #118
OCCURS-IN
RESOURCE/LOCATION
STACK-BASED #121
ARRAY #129
HEAP-BASED #122
MEMORY-BUFFER #119
INDEX (POINTER #466 INTEGER #129)
PART-OF
BUFFER #119
INDEXABLE-RESOURCE #118
PART-OF
IS-A
CONSEQUENCES
UNCONTROLLED MEMORY ALLOCATION #789
WRITE-WHAT-WHERE CONDITION #123
INFORMATION LOSS OR OMMISSION #199 #221
CAN-PRECEDE

distinct weakness type have been developed. The templates can then be readily applied to aggregate and study project-specific vulnerability information from source code repositories.

Another approach to improve the CWE is Software Fault Patterns (SFPs) [8]. SFPs decompose CWEs by fine granularity patterns with white-box definitions, then compose them into original CWEs with invariant core and variation points. With the purpose of being integrated into a standards-based tool analysis approach, SFPs focus more on the source code faults and the features that can facilitate automation. Such automation can potentially be very valuable for software assurance activities described in [12], because CWE has an important role in those activities [12].

4.1 Semantic Templates

A Semantic Template is a human and machine understandable representation that contains the following four elements [5]:

1) Software faults that lead to a weakness
2) Resources that a weakness affects
3) Weakness characteristics
4) Consequences/failures resulting from the weakness.

The required information pieces are either expressed together within a single CWE entry or spread across multiple entries. Such complexity makes it difficult to trace the information expressed in the CWE to the information about a discovered vulnerability from multiple sources. Therefore, to facilitate CWE use in the study of vulnerabilities, easy-to-understand templates for each conceptually distinct weakness type have been developed. These templates can then be readily applied to study project-specific vulnerability information from project repositories. For example, figure 2 shows the Semantic Template for Buffer Overflow, which is an aggregation of information collected from 42 CWEs. In this Buffer Overflow Semantic Template, the four groups of relevant information were carefully collected and synthesized with "is-a" relationship inside of each group and "can-precede", "occurs-in" between the groups so that the lifecycle of a weakness from the starting point (software fault) to the end (consequences) is clearly presented.

The Semantic Templates also can provide intuitive visualization capabilities for the collected vulnerability information such as the CVE vulnerability descriptions, change history in the open source code repository, source code versions (before and after the fix), and related CAPECs [6]. Semantic Templates were shown to be helpful to programmers in constructing mental models of software vulnerabilities by an experiment described in [7]. In this experiment, 30 Computer Science students from a senior-level undergraduate Software Engineering course were selected to study six sets of vulnerability-related material with or without Semantic Templates in a pre-post randomized two-group design. The experimental results revealed that the group with the aid of Semantic Templates could analyze vulnerabilities with shorter time and higher recall on CWE identification accuracy.

4.2 Software Fault Patterns

Software Fault Patterns (SFPs) was developed by KDM Analytics Inc. By identifying and developing white box definitions for SFPs as a formalization process, they could be integrated into a standards-based tool analysis approach, benefiting both real-time embedded and enterprise software assurance systems. Those identified SFPs will be common to more than one CWE and can be used to further define CWEs [8].

The SFP is targeted at preventing cyber-attacks by collecting and managing knowledge about exploitable weaknesses and building more comprehensive prevention, detection and mitigation solutions. With the knowledge extracted from CWE taxonomy, three transformations were executed to extract common patterns and white-box knowledge, redefine existing weaknesses as specializations of the common patterns, then invariant core and variation points are identified to redefine each SFP to further represent weakness specializations [8].

KDM Analytics defines an SFP as a common pattern with one or more associated pattern rules (conditions), representing a family of faulty computations. The SFP structure is organized by the primary SFP definition which refers to the entire secondary cluster and is arranged into invariant core and variation points [8]. SFPs can map to multiple CWEs in such a way that each CWE in the family can be defined as a specialization of the SFP with its specific variations on the identified parameters. To date, 21 primary clusters, which include totally 62 secondary clusters, and 36 unique SFPs have been identified. 632 CWEs have been categorized while only 310 of them are identified as discernible CWEs. Identified SFP definitions could lead to the development of more accurate testing tools and also improve developer education and training. They also provide benefits for a possible future formalization, since for each CWE, only the variation extension to a formalized SFP is required.

As the proof of recognition of the SFP research work, CWE-888: Software Fault Pattern (SFP) Clusters was incorporated by MITRE as a view into the CWE dictionary.

Both Semantic Templates and SFPs are designed to help understand and automate the vulnerability study. While Semantic Templates emphasize mental model construction from the human perspective, with the explanation of the four main elements of a vulnerability's lifecycle, while SFP's approach focuses on the "foot-holds", which are places in the code that present the necessary conditions for vulnerabilities, with the emphasis on the computation side to aid the test cases generator's work.

5. Future Directions on Improving CWE

This section provides future directions and our vision on CWE formalization.

CWE is a unique community effort and already has been proved to be extremely useful. For example, the NIST SAMATE project has utilized CWE during the past four Static Analysis Tool Expositions (SATE), whose goal is to advance research in static analysis tools that look for security defects in source code [9]. CWE is "a unifying language of discourse and a measuring stick for comparing tools and services" [10]. It is used in a wide variety of domains by developers and testers to look for known weaknesses in the code, design, and architecture of their software products; by consumers to make informed decisions when selecting software security tools and services; by researchers to develop new approaches and tools for software testing; and by professors to teach software developers how to avoid known weaknesses on architecture, design, and code level, in order to avoid security problems on applications, systems, and networks.

CWE is meant to be "a formal" list of software weakness types [1]. However, the CWE descriptions are currently in natural language and sometimes not accurate or precise by using phrases such as "correctly perform," "intended command," "intended boundary." For example, the description summary of CWE-119 in http://cwe.mitre.org/data/definitions/119.html includes the term "intended boundary", which is too vague. It does not indicate that it is the boundary given by the formal semantics.

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

"The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer."

While, to mitigate the vagueness of the definition as much as possible, our tentative definition of CWE-119 is: The software can access through a buffer a memory location not allocated to that buffer [11].

Therefore, the next logical step is to formalize CWE definitions, as formal approaches are less ambiguous and offer high level of accuracy. Our vision for CWE formalization and creating a system of accurate, precise definitions of CWEs, although a high-bar, is as follows:

- Revamp CWE entries towards Software Fault Patterns
- Review for accuracy existing CWE description summaries and white-box descriptions
- Analyze descriptions meaning and remove ambiguities
- Precisely define CWE entries with required accuracy
- Decide on a formal specification language
- Formalize CWE definitions
- Determine approach for validating CWE definitions
- Determine approaches for automated generation of tools for validation and verification towards particular weaknesses.

It is challenging to identify known weaknesses as well as newly discovered weaknesses, but it is challenging also to describe them in a succinct and unambiguous manner. Formalization should come in place and help further "shape and mature the code security assessment industry and dramatically accelerate the use and utility of automation-based assessment." [1]

Semantic Templates builds on CWE, and introduces a novel reorganization of CWE. One example for a potential use of Semantic Templates is for automatic change analysis. Patches provided by contributors to open source software may introduce vulnerabilities. Semantic Templates may help in organizing knowledge about known vulnerabilities in a way that will help patch contributors to detect vulnerabilities [5] .

Once formalized the CWE definitions could be easily expressed through formal description techniques (FDT) and used as an input for generation of testing codes. This would facilitate automatic generation of more precise CWE-compatible software analysis and profiling tools for discovery of vulnerabilities or prioritizing vulnerabilities in terms of threats and impacts. Especially valuable would be the application for generation of dynamic analysis tools, which are better at discovering run-time vulnerabilities that cannot be captured with static-code analysis techniques – for example, buffer overflow lends itself to such dynamic analysis.

6. Conclusion

CWE provides common terminology for software developers, security experts, researchers, and customers to discuss software vulnerability in design, systems architecture, and source code. Software is central to computer science and as one of the purposes of CWE is to help avoid and eliminate software flaws in various stages of software production, CWE is of value not only to the software assurance community, but to computer science as a whole.

Improving quality of software development to reduce instances of weaknesses takes work from language designers, compiler writers, educators, assurance tool developers, researchers, vulnerability trackers, software engineers, and many more. If people in these roles disagree about what constitutes a particular weakness, or even whether it is a weakness at all, communication would be difficult at best. Therefore, broadly accepted definitions should be developed to allow diverse groups to work effectively together. It is important the definitions to be unambiguous and complete to allow professional in the field to understand precisely what different software assurance tools, services, technologies, or methods can detect, mitigate, or prevent. Pure formalization of CWE would allow automatic generation of software components and tools to test for weaknesses that lead to exploitable vulnerabilities in software, create wrappers to filter out attacks that exploit them, or even rewrite the code to eliminate them.

Once precisely defined, CWEs could be formally described using a specification language such as Alloy (http://alloy.mit.edu/alloy). At its core, Alloy has a simple but expressive logic based on the notion of relations. Its syntax is designed to make it easy to build models incrementally and it has a rich sub-type facility for factoring out common features and a uniform and powerful syntax for navigation expressions.

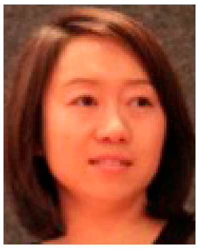
To provoke further thinking and discussions throughout the Software Assurance community and beyond, we pose the following questions:

- What other formal methods can be used to help formalize CWEs with required accuracy and precision and at the same time allow for further extensions?
- To what granularity should CWEs be formalized? Finer granularity means more flexibility (especially when new weaknesses are identified, the extracted commonalities can reduce the re-invent work) but more effort to create them; Coarser granularity indicates the easy-to-use weakness items while we need to re-invent the wheel every time.
- How can the formalized CWEs be used and in which domains? For education and training? To prevent vulnerabilities? To integrate into software IDEs, test tools, and tools that generate test tools? To integrate in application security and development security technical implementation guides such as that of DOD [13].
- How can an automatic system be constructed to record newly identified vulnerabilities and classify them by CWEs? With better formalization and finer granularity of CWE definitions (which also means limited dictionary for weaknesses, better taxonomy of vulnerabilities), text mining could be the potential technique to mapping CVEs to CWEs at least semi-automatically.

REFERENCES

1. MITRE. “CWE Common Weakness Enumeration.” http://cwe.mitre.org
2. MITRE. “CVE Common Vulnerabilities and Exposure.” http://cve.mitre.org
3. MITRE. “CWE Common Weakness Enumeration Common Weakness Scoring System (CWSS) CWSS Version 0.8.” June 2011. Project Coordinator: Bob Martin, Document Editor: Steve Christey. http://cwe.mitre.org/cwss
4. MITRE. “Common Attack Pattern Enumeration and Classification (CAPEC)TM A Community Knowledge Resource for Building Secure Software.” http://makingsecuritymeasurable.mitre.org/docs/capec-intro-handout.pdf
5. Y. Wu, R. A. Gandhi, and H. Siy. “Using semantic templates to study vulnerabilities recorded in large software repositories.” 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS ‘10, pages 22-28, New York, NY, USA, 2010. ACM.
6. R. Gandhi, H. Siy, Y. Wu. “Studying Software Vulnerabilities.” CrossTalk, The Journal of Defense Software Engineering, September/October 2010.
7. Y. Wu, H. Siy, R. Gandhi. “Empirical Results on the Study of Software Vulnerabilities (NIER Track).” 33rd International Conference on Software Engineering (ICSE 2011), Honolulu, Hawaii. May 2011.
8. B.A. Calloni, D. Campara, and N. Mansourov. (2011). Embedded Information Systems Technology Support (EISTS) ---Task Order 0006: Vulnerability Path Analysis and Demonstration (VPAD), Volume 2 - White Box Definitions of Software Fault Patterns.
9. NIST. “Special Publication 500-297 Report on the Static Analysis Tool Exposition (SATE) IV.” http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-297.pdf
10. R. Martin, S. Barnum, S. Christey. “Being Explicit about Security Weaknesses” http://cwe.mitre.org/documents/being-explicit/BlackHat_BeingExplicit_WP.pdf
11. Paul E. Black, Yan Wu, Yaacov Yesha, Irena Bojanova, in preparation.
12. Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)) and Department of Defense Chief Information Officer (DoD CIO). Software Assurance Countermeasures in Program Protection Planning. Washington, D.C. 2014. www.acq.osd.mil/se/docs/SwA-CM-in-PPP.pdf
14. DISA for DOD, Application Security and Development Security Technical Implementation Guide (STIG), Version 3, Release 8. 25 July2014, http://iase.disa.mil/stigs/app-security/app-security/Pages/index.aspx
15. MITRE: CWE Common Weakness Enumeration, Frequently Asked Questions (FAQ), http://cwe.mitre.org/about/faq.html#A.8

ABOUT THE AUTHORS

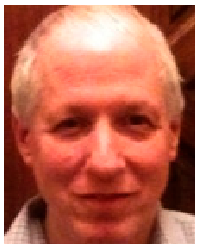


Yan Wu is currently working as an assistant professor at Computer Science Department of Bowling Green State University, and she previously was a guest researcher in SAMATE team at NIST. She received her Ph.D. degree in Information Technology in 2011 from the University of Nebraska at Omaha. The main goal of her research is to conduct empirical study on analyzing software engineering knowledge in order to support the development and maintenance of reliable software-intensive systems.

E-mail: yanwu@bgsu.edu

Irena Bojanova is a professor and program director of Information and Technology Systems at UMUC. She is the founding chair of the IEEE CS Cloud Computing STC, a general chair of the IT Professional Conference <http://tinyurl.com/itproconf> , and coeditor of Encyclopedia of Cloud <http://tinyurl.com/EncyclopediaCC>
Computing (Wiley, to appear in 2014). She is also an associate editor in chief of IT <http://www.computer.org/itpro>
Professional and an associate editor of IEEE Transactions on Cloud Computing <http://www.computer.org/portal/web/tcc> . You can read her cloud computing blog at www.computer.org/ portal/web/Irena-Bojanova.

E-mail: irena.bojanova@umuc.edu



Yaacov Yesha is a Professor at the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County. He received his PhD in Computer Science in 1979 from the Weizmann Institute of Science. He has received substantial research funding from government and industry. He was a program committee member of several conferences and a Chair of two workshops at IBM CASCON 2007.

E-mail: yayesha@cs.umbc.edu

Upcoming Events

Visit ‹http://www.crosstalkonline.org/events› for an up-to-date list of events.

2015 7th International Conference on Software Technology and Engineering (ICSTE 2015)
September 19-20, 2015
Hong Kong
http://www.icste.org/

SEDE 2015: 24th International Conference on Software Engineering and Data Engineering
Oct 12-14, 2015
San Diego, Ca
http://www.cse.unr.edu/SEDE

STC 2015, the 27th Annual IEEE Software Technology Conference
October 12 - 15, 2015
Long Beach, CA
http://conference.usu.edu/STC

EnCASE 2015: Second International Workshop on Engineering Cloud Applications & Services
Oct 19, 2015 – Oct 21, 2015
Rome, Italy
http://www.dis.uniroma1.it/~soca2015

SLE 2015: ACM SIGPLAN Software Language Engineering
Oct 25-27, 2015
Pittsburgh, PA
http://conf.researchr.org/home/sle2015

SEMCMI 2015: The International Conference on Software Engineering, Mobile Computing and Media Informatics- Part of the Fourth World Congress on Computing and Information Technology
Kuala Lumpur
October 27-29, 2015
http://sdiwc.net/conferences/semcmi2015

Better Software Conference East
Nov 8- Nov 13, 2015
Orlando, Florida
http://bsceast.techwell.com

30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)
November 9-13, 2015
Lincoln, Nebraska
http://ase2015.unl.edu/#tab-main

ACTION15: Actionable Analytics for SE
Nov 9, 2015 – Nov 13, 2015
Lincoln, Nebraska
http://action15.github.io

2015 IEEE 23rd International Conference on Network Protocols (ICNP)
Nov 10, 2015 – Nov 13, 2015
San Francisco, CA
http://icnp15.cs.ucr.edu

INFuture2015: e-Institutions – Openness, Accessibility, and Preservation
Nov 11, 2015 - Nov 13, 2015
Zagreb, Croatia
http://infoz.ffzg.hr/INFuture

2015 SC - International Conference for High Performance Computing, Networking, Storage and Analysis
Nov 15, 2015 – Nov 20, 2015
Austin, TX
http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=32761

CrossTalk
The Journal of Defense Software Engineering
Vol. 26 No. 1
Jan / Feb 2013
SOFTWARE PROJECT MANAGEMENT
LESSONS LEARNED
July / August 2011
Vol. 24 No. 4
Vol. 24 No. 2
Jan / Feb 2012
Vol. 25 No. 5
March / April 2013
Vol. 24 No. 1
Nov / Dec 2010
Vol. 23 No. 6
Enjoy!

Supply Chain UnAssurance
An Alternate View of History by Dave Cook

"An army marches on its stomach, but sits on its ……"

I know the question that you are dying to ask is, "Who is the father of preserved foods," right? Clarence Birdseye? Nope – sorry – he was the father of flash-frozen foods. The honor of being the father of preserved foods belongs to Nicolas Appert.

Appert was a confectioner and chef in Paris from 1784 to 1795. In 1795, he began experimenting with ways to preserve foodstuffs, succeeding with soups, vegetables, juices, dairy products, meats, jellies, jams, and syrups. He placed food in glass jars, sealed them with cork and sealing wax and placed them in boiling water.

Why would a confectioner develop such a fixation on preserving foods? Well, you see, it was not about a fixation, it was about money! France was engaged in various conflicts during the late 1790s and early 1800s. The French Army was well aware that an army marches on its stomach. When engaged in conflicts in locations where the locals were unable (or unwilling) to supply provisions, it was difficult to keep the soldiers well fed.

So, in 1795 the French military offered a cash prize of 12,000 francs (about about USD $2 million in today's money) for a new method to preserve food. Appert was definitely interested! After some 14 or 15 years of experimentation, Appert submitted his invention and won the prize in January 1810 on the condition that he make the method public. So, the same year, Appert published *L'Art de conserver les substances animales et végétales* (or *The Art of Preserving Animal and Vegetable Substances*). This is considered the world's first cookbook concerning modern food preservation methods. His method was to cut up and place foodstuffs in a bottle, leaving air space at the top of the bottle. Then, a cork would then be sealed firmly in the jar by using a vise, and the bottle was then wrapped in canvas to protect it. Next, the bottle was placed into boiling water and then boiled for as much time as Appert deemed appropriate for cooking the contents thoroughly. The cork was then reinforced with wire.

It was not a fast process (it took up to five hours per bottle), but it worked. It is worth noting that Appert (nor anybody else at the time) had any idea WHY it worked – it would be almost 50 years before another Frenchman, Louis Pasteur, showed the relationship between bacteria and food spoilage. In any case, Appert was given the prize, and started a company that produced canned foodstuffs for more than 100 years.

Which leads us to 1815. Napoleon at this time was self-proclaimed Emperor, and was engaged against the coalition armies of Great Britain, Russia, Austria, and Prussia at the battle of Waterloo.

What in the world does food preservation have to do with Waterloo? You see, the majority of the foodstuff that the French carried with them to battle consisted of energy-giving proteins – beef, lamb, and other meats. I don't know about your digestive system, but a diet heavy in protein (and low in high-fiber food choices) can make you wish that there were large supplies of prunes also accompanying the meal. During the Waterloo conflict, Napoleon was troubled by hemorrhoids, which made sitting on a horse for long periods of time difficult and painful.

This condition led to disastrous results at Waterloo. Waterloo occurred on Sunday, 18 June 1815, near Waterloo in present-day Belgium. There had been several days of fighting – on the previous Friday, the French defeated the Prussian army at Ligny (about 20 miles from Waterloo). This turned out to be the last battle ever won by Napoleon.

On Sunday, the British troops, led by the Duke of Wellington, combined with the re-grouped Prussians (led Gebhard von Blücher) and attacked. During the critical phases of the battle, Napoleon was unable to sit on his horse for other than very short periods of time. This greatly interfered with Napoleon's ability to survey the situation and direct his troops effectively. Napoleon was unaware just how weak his right flank had become – and the Prussians were able to break through the weakened right flank.

At the same time, Wellington attached from the front – and the French were driven from the battlefield in complete disarray. All because Napoleon couldn't sit in his saddle. Probably due to a high-protein low-fiber diet. Because food preservation techniques did not require the French to forage for local produce (where the local diet would have been healthier and higher in fiber).

Maybe a few cases of prunes or some Milk of Magnesia would have helped. But then, those weren't available in the supply chain.

The supply chain – like software – needs to know not just the requirements, but also the actual needs of its users. But that's another Backtalk column.

David A. Cook
Stephen F. Austin State University

CrossTalk / 517 SMXS MXDED
6022 Fir Ave.
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737

Homeland Security

Software and Supply Chain Assurance

Software and Supply Chain Assurance are essential to enabling the Nation's critical infrastructure.
To ensure the integrity of that infrastructure, the software and the IT supply chain must be secure and resilient.

The Software Assurance Community Resources and Information Clearinghouse provides corroboratively developed resources. Visit https://buildsecurityin.us-cert.gov/swa/index.html to learn more about relevant programs and how you can become involved.

Software and Supply Chain Assurance must be "built in" at every stage of development and supported throughout the lifecycle.
Visit https://buildsecurityin.us-cert.gov/bsi/home.html to learn about the practices for developing and delivering software to provide the requisite assurance. Sign up to become a free subscriber and receive notices of updates.

The Department of Homeland Security provides the public-private collaboration framework for shifting the paradigm to software and supply chain assurance.

Software Assurance
Community Resources and Information Clearinghouse
Sponsored by DHS National Cyber Security Division

HOME | ABOUT | RESOURCES | ADVISORIES | EVENTS | WEBINARS | PODCASTS | PROCESS VIEW

SwA Communities
SwA Forums & Working Groups
Workforce Education & Training
Processes & Practices
Technology, Tools & Product Eval.
Acquisition & Outsourcing
Measurement
Business Case
Malware
SwA Landscape
SwA Ecosystem
Build Security In

Software Assurance

Announcements and Current Activities

- Software Assurance Curriculum Materials Available for Download
- SwA Ecosystem Page Added
- FAQs Added on SwA Forums and Working Groups
- New Software Assurance Self Assessment Effort Resource Available from the Processes & Practices Working Group

Learn About...

- SwA Forums & Working Groups, composed of stakeholders in government, in and academia, who are contributing to efforts focused on advancing softwar assurance objectives and evolving the Assurance Ecosystem.
- Upcoming SwA Forums and Work Group sessions
- SwA Pocket Guide Series, free, downloadable documents on critical software assurance topics
- Mitigating the Most Exploitable Software Weaknesses: Top 25 CWE Programm Errors

Contact Us Terms of Use Privacy Policy Sitemap Get a PDF Reader
©2010 Department of Homelan

Build Security In
Setting a higher standard for software assurance
Sponsored by DHS National Cyber Security Division

Homeland Security
Privacy and Use

Navigational Links
Home
Mission
Articles [by Content Area]
Events
About Us
FAQs
Secure Coding Sites
Additional Resources
DHS SwA Web Site
DHS Software Assurance Resources
RSS Feeds
Contact Us

Build Security In Home

What is Build Security In?

Build Security In is a collaborative effort that provides practices, tools, guidelines, rules, principles, and other resources that software developers, architects, and security practitioners can use to build security into software in every phase of its development.

Introduction to Software Security

The Software Assurance Curriculum Project

This project provides materials for undergraduate and master's courses in software assurance. Materials are now available for download.

Call for Reviewers

Volunteer to review new articles for publication on BSI. See the Call for Reviewers for details.

Community Collaboration

To access other software assurance materials or to join the collaboration efforts of a related working group, visit the DHS Software Assurance Program's Community Resources and Information Clearinghouse.

Sponsor and Contributors

Build Security In is a Software Assurance

Improve Security and Software Assurance: Tackle the CWE Top 25 – The Most Dangerous Programming Errors

The Top 25 CWEs represent the most significant exploitable software constructs that have made software so vulnerable. Addressing these will go a long way in securing software, both in development and in operation. Read more and see the list of Top 25 CWE Programming Errors on the Software Assurance Community Resources and Information Clearinghouse website.

Consistent with this list is the Top 10 for 2010 released by the Open Web Application Security Project (OWASP). OWASP's report captures the top ten risks associated with the use of web applications in an enterprise. Download the report, which contains examples and details that explain these risks to software developers, managers, and anyone interested in the future of web security, for free here.

What's New

A new area, CrossTalk Series, has been added to Additional Resources.

Three new articles, MITRE CWE and CERT Secure Coding Standards, An Evaluation of Cost-Benefit Using Security Requirements Prioritization Methods, and Improving Software Assurance, have been added.

A new area for the Software Assurance Curriculum Project has been added. The Master of Software Assurance Curriculum and Undergraduate Course Outlines are now available.

Calls for papers have been posted for the 1st International Workshop on Measurability of Security in Software Architectures (MeSSa), and the Software Assurance Minitrack of the 44th Hawaii International Conference on System Sciences (HICSS-44).

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

CrossTalk thanks the above organizations for providing their support.