SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

The Software Supply Chain Integrity Framework

Defining Risks and Responsibilities for Securing Software in the Global Supply Chain

July 21, 2009

Editor
Stacy Simpson, SAFECode

Contributors
Dan Reddy, EMC
Brad Minnis, Juniper Networks
Chris Fagan, Microsoft Corp.
Cheri McGuire, Microsoft Corp.
Paul Nicholas, Microsoft Corp.
Diego Baldini, Nokia
Janne Uusilehto, Nokia
Gunter Bitz, SAP
Yuecel Karabulut, SAP
Gary Phillips, Symantec

Table of Contents

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

Introduction

Commercial software underpins the information technology infrastructure that businesses, governments and critical infrastructure owners and operators rely upon for even their most vital operations. For that reason, enterprise customers are rightfully concerned about the security of commercial software and the potential for its exploitation by those seeking to maliciously disrupt, influence or take advantage of their operations.

Historically, commercial software was developed at a central location. However, as market demands for innovation and competitiveness have increased, a more distributed approach to software development is evolving as commercial software vendors expand to serve international markets and seek engineering skills and numbers wherever they reside globally.

Though it is generally agreed that limiting the use of global resources for software development is not practical in today's market environment, the increased distribution of development activities globally does raise questions about what additional product security and commercial brand risks are introduced, how these risks should be assessed, and what proactive measures can minimize their occurrence. Given the reliance of businesses, governments and critical infrastructure owners on commercial software, these questions are of interest to suppliers and customers alike and have recently been aggregated under the label of "software supply chain integrity."

Yet, the concept of software supply chain integrity and its key components of "software integrity" and "software supply chain" are not clearly defined, creating significant challenges for customers and suppliers working to identify, compare, communicate and evaluate software integrity best practices.

Recognizing this gap, SAFECode is working to address the issue of software supply chain integrity from a software engineering perspective. Under this effort, SAFECode will identify the threats, assess the risks, share its members' current practices for mitigating those corresponding risks, and develop process guidelines that other software companies should consider adopting to protect the integrity of the software they produce through the global supply chain.

This paper, the first in a series, will assess software supply chain integrity in the context of software engineering, providing a framework and common taxonomy for evaluating the associated risks and defining the industry's role in addressing them. This framework will serve as the foundation for subsequent work aimed at describing and analyzing software integrity best practices.

Defining Software Integrity

Software integrity is an element of software assurance. SAFECode defines Software Assurance as "confidence that software, hardware and services are free from intentional and unintentional vulnerabilities and that the software functions as intended."[1]

Software assurance is most frequently discussed in the context of ensuring that code itself is more secure through the repeatable application of secure software development practices. However, while there has been a growing and appropriate focus on eliminating software vulnerabilities through secure development practices, this represents only one aspect of software assurance. Another key consideration for customers and software suppliers is the security of the processes used to handle software components during their sourcing, development and distribution since a variety of potential attack vectors exist throughout the software lifecycle.

To help others in the industry initiate or improve their own secure development programs, SAFECode has published "Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today." Based on an analysis of the individual software assurance efforts of SAFECode members, the paper outlines a core set of secure development practices that can be applied across diverse development environments to improve software security.

The brief and highly actionable paper describes each identified security practice across the software development lifecycle – Requirements, Design, Programming, Testing, Code Handling and Documentation – and offers implementation advice based on the real-world experiences of SAFECode members.

To obtain a free copy of the paper, visit **www.safecode.org**.

1. SAFECode, "Software Assurance: An Overview of Current Best Practices," February 2008.

In practice, software assurance involves a shared responsibility among suppliers (synonymous with vendors), service and/or solution providers, and customers encompassing three areas:

- **Security:** Security threats are anticipated and addressed in the software's design, development and testing. This requires a focus on both quality aspects (e.g., "free from buffer overflows") and functional requirements (e.g., "passport numbers must be encrypted in the database").
- **Authenticity:** The software is not counterfeit and customers are able to confirm that they have the real thing.
- **Integrity:** The processes for sourcing, creating and delivering software contain controls to enhance confidence that the software functions as the supplier intended.

While the delivery of secure software products requires that all three elements of software assurance be addressed, the focus of this paper is on software integrity practices – the collection of processes and controls that enable a supplier to deliver a product to customers that is uncompromised, thereby containing only what the supplier intends. Software integrity practices address the security of the processes used to handle software components during their sourcing, development and delivery. In this way, they differ from (and complement) secure development practices that improve the security characteristics of the code comprising the software components. Software integrity practices are essential to minimizing the risk of software tampering in the global supply chain.

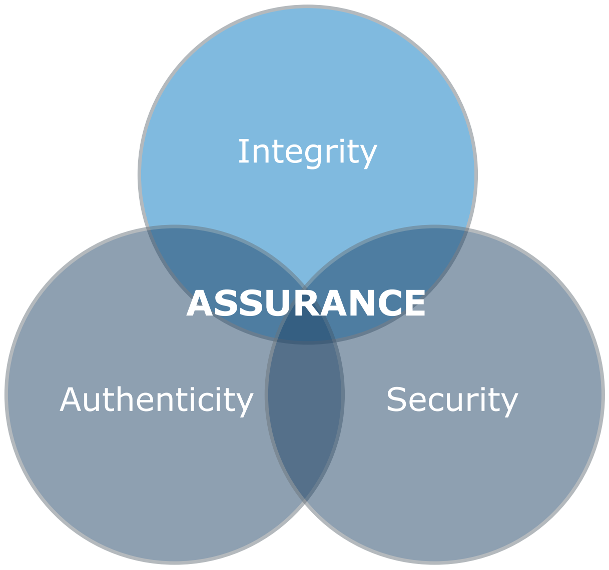

Figure 1: The three elements of software assurance

Identifying the Challenge to Software Integrity

Governments, businesses and consumers purchase IT solutions (systems, products or services) that are a complex collection of inter-related components assembled from hardware, software, networks, cloud services and outsourced operations. Throughout an IT solution's lifecycle, which can extend over more than a decade, many individuals have legitimate access to its components and operations.

As the software industry has become increasingly globalized, a concern has risen over the possibility that an IT solution could be compromised by the intentional insertion of malicious code into the solution's software during its development or maintenance. This type of attack is often referred to as a supply chain attack. A supply chain attack can be directed at any category of software, including custom software, software delivering a cloud service, a software product, or software embedded in a hardware device.

Software in any of these categories is often packaged as a collection of files. To be successful, a software supply chain attack must result in either: a) the modification of an existing software file(s); or, b) the insertion of an additional file(s) into the collection of software files.

Reports[2] that have considered supply chain attacks have concluded that: 1) there is no one way to defend against all the potential attack vectors a motivated attacker may identify; 2) focusing on the place where software is developed is less useful for improving security than focusing on the process by which software is produced and tested; and 3) there are circumstances when the insertion of malicious code would be almost impossible to detect.

It is important to recognize that while there is a risk that someone with malicious intent could attack software during its development, experts[3] have concluded that supply chain attacks are not the most likely attack vector. For example, the practice of hackers or other malicious actors finding and exploiting existing vulnerabilities remains the most common method of attack.

2. "Mission Impact of Foreign Influence on DoD Software," U.S. Defense Science Board, September 2007. "Foreign Influence on Software: Risk and Recourse," Center for Strategic and International Studies, March 2007. "Framework for Lifecycle Risk Mitigation For National Security Systems in the Era of Globalization," U.S. Committee on National Security Systems, November 2006.

3. "Mission Impact of Foreign Influence on DoD Software," U.S. Defense Science Board, September 2007. "Foreign Influence on Software: Risk and Recourse," Center for Strategic and International Studies, March 2007.

However, the fact that a risk does exist requires preventive action. While individual software companies have taken steps to assure the integrity of their own supply chains, there is currently no framework or shared taxonomy through which the software industry can collectively identify and develop best practices to address software supply chain threats to software integrity.

Further, the software supplier has a dual challenge. As the vendor offering a product, the customer often views the supplier as responsible for all the offering's components. That supplier is also an acquirer of software components. As such, a supplier must ascertain the integrity (along with authenticity and security) of both the software components they build and those that they acquire or use to be well-positioned to assert integrity claims for their product.

Describing the Software Supply Chain

Sophisticated IT solutions have much in common with other engineering undertakings. Each IT solution is a collection of components. Each component or its parts can be a) developed by its supplier or on that supplier's behalf by their subcontractors; or b) licensed to the supplier by another vendor or obtained from Open Source repositories; or c) acquired outright by the supplier.

However, this complexity of components within components can be organized. In the physical world many industries create complex products that contain components from multiple sources. Processes in the manufacturing of physical goods have two parallels that can be adopted in the cyber world. One is the use of a Bill of Materials (BOM) to organize the hierarchy of product components. The other is the use of supply chain management processes, which describe the business activities associated with satisfying a customer's demand spanning the range from the supplier's supplier to the customer's customer.

By recognizing and adapting techniques pioneered for the physical world, IT suppliers can identify natural control points within software supply chains. To identify these points, consider that each software supplier has three links of the supply chain. For these three links each IT supplier takes similar actions:

1. **Supplier Sourcing:** Select the suppliers, establish the specification for the supplier's deliverables, and receive software/hardware deliverables from the suppliers;
2. **Product Development and Testing:** Build, assemble, integrate and test components and finalize for delivery; and,
3. **Product Delivery:** Deliver and maintain their product components to their customer.

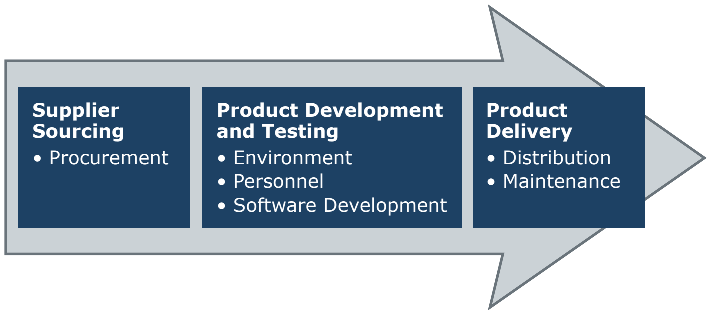

Figure 2: Each supplier in the software supply chain manages three sets of controls

Now consider that delivered software is just one component of a larger IT solution and each software supplier is only one vendor in a complex chain of suppliers and systems integrators. Customer relationships extend even beyond the traditional system integrators since some "acquirers" implement systems as solutions for other end users. As such, the software supply chain is only one part of a larger, more complex IT solution supply chain.

Figuratively, an IT solution supply chain resembles a collection of staircases. Each staircase aggregates smaller useful components from different suppliers into an ever-greater collection of components until ultimately sufficient IT components have been assembled to meet the customer's business requirements.

Figure 3 illustrates the software supply chain as one of these staircases, where each step holds a different supplier. Between each step is a step-up. The step-up represents the transmission of software components from a supplier to its customer.

Components move along this "staircase" supply chain as they are handed off from one supplier to the next. At each step a supplier controls three links in the supply chain: a) goods received from suppliers; b) their product production; and c) what is delivered to their customers. Suppliers apply integrity controls at each link. For example, a supplier can conduct acceptance tests on components received from their suppliers, and release tests on the components they deliver to their customer. That means that each transition of custody along the chain of suppliers is an opportunity to preserve code integrity.

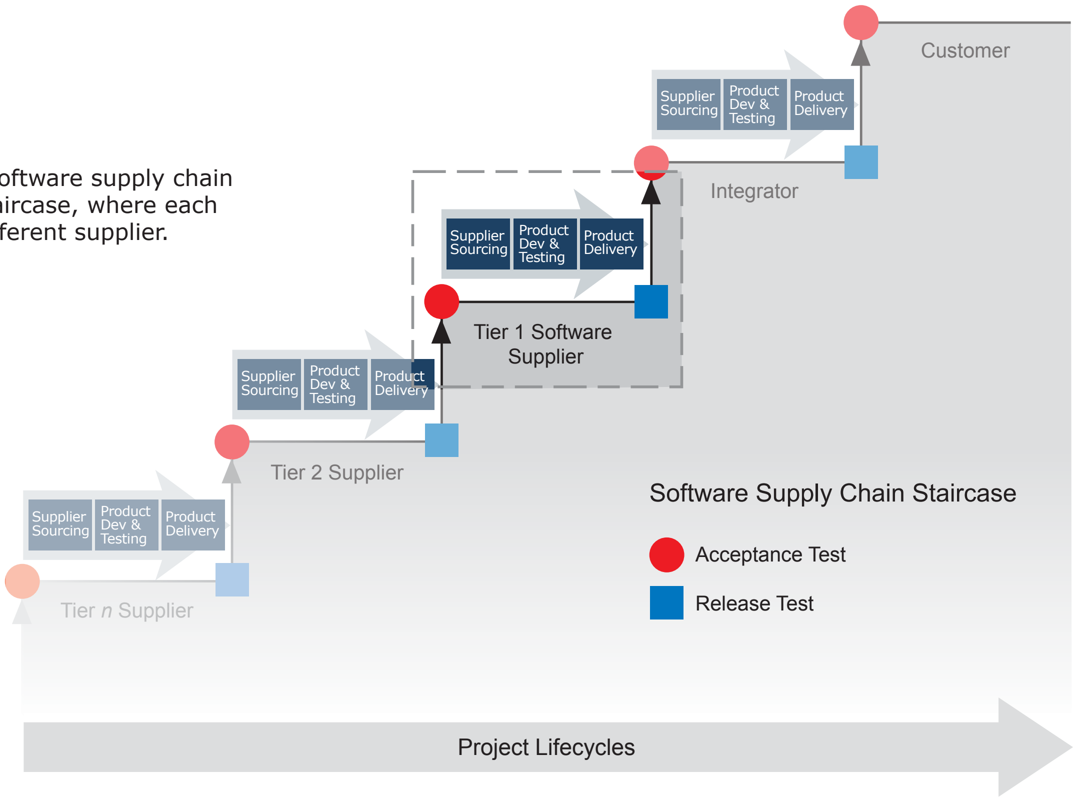

Figure 3: The software supply chain resembles a staircase, where each step holds a different supplier.

The effective application of integrity controls requires that each individual supplier understand and manage:

- The components that are integrated into their products. This includes identifying their suppliers and the related parties including, for example, software from original equipment manufacturers (OEMs), software built to specification by external contractors, or sourced from repositories of Open Source Software (OSS);
- Their internal processes for controlling access to software components during development, integration, testing and release; and,
- The channels they use to receive components from suppliers and to deliver products.

Within a supplier's organization, operations for supplier sourcing, product development and testing, and product delivery coexist with other business operations such as sales, marketing, legal and IT. The procedures a supplier uses for internal collaboration, distribution and other important business processes outside product development and testing can impact a product's integrity. Concern about personnel with malicious intent is not confined to only employees at numerous locations, but extends across all members of the workforce including that of tiered suppliers.

Other factors must come into play to address integrity. The type of physical and IT environment supporting product development and testing has a bearing on the likelihood of a product's integrity being compromised. For instance, are the facilities where code is developed secure? Is the data center where code is stored secure? Are communications secure between distributed teams?

Within a supplier, different business functions are performed by different staff, and these personnel require different levels of access to corporate assets. Access to corporate assets is based on the security principle of separation-of-duties. This principle should be similarly applied to access to development assets by a supplier's staff engaged in supplier sourcing, product development and testing, and product delivery. Access to development assets can be restricted based on the activities a staff member performs in the development process.

Additionally, suppliers control: a) how they procure code from their suppliers; b) how code once delivered is screened and tested; and c) how code is inserted and tracked throughout their development, testing and delivery processes. While the effective application of

these controls has a direct impact on software integrity, it should be recognized that leading software suppliers also establish and maintain these controls for sound engineering reasons since the production of commercial software products is an industrial-strength process.

To assist customers, suppliers provide mechanisms to enable validation of a product's authenticity, and to confirm that a product has not been tampered with before it reaches them. These may include certificates of authenticity, online product registration and validation, and product packaging designed to be tamper-resistant.

Having recognized these issues, suppliers have controls over their software products when components are received from their suppliers, created through their product development process, and passed on to their customers. It is this collection of controls that enable a supplier to assure that the software components they use are known and properly protected along the supply chain.

Principles for Designing Software Integrity Controls

Viewing potential malicious acts in the proper context is essential. Suppliers are aware of threats to their products and are, consequently, extremely protective of their code base – not only is the integrity of their products at stake but also their highly valuable intellectual property and brand. As such, suppliers have significant experience implementing powerful management, policy and technical controls that reduce the risk that their code can be compromised. However, while there are established practices to mitigate the potential for malicious activity as products are built, analyzing these efforts in the context of providing assurance of integrity is an emerging discipline. As a result, there is a lack of common understanding regarding what these efforts entail, where they are best applied in the context of a software supply chain, and how collectively they raise the assurance of an IT solution.

Within SAFECode's software supply chain integrity framework, software supply chain integrity controls address the access, storage and handling of development assets throughout the key links in the software supply chain – supplier sourcing, product development and testing, and product delivery. Within a supplier's organization these controls exist in the context of other IT functions such as backup and recovery, business continuity services, physical and network security, and configuration management systems.

Software supply chain integrity controls derive from established security and integrity principles:

- **Chain of Custody:** The confidence that each change and handoff made during the source code's lifetime is authorized, transparent and verifiable.
- **Least Privilege Access:** Personnel can access critical data with only the privileges needed to do their jobs.
- **Separation of Duties:** Personnel cannot unilaterally change data, nor unilaterally control the development process.
- **Tamper Resistance and Evidence:** Attempts to tamper are obstructed, and when they occur they are evident and reversible.
- **Persistent Protection:** Critical data is protected in ways that remain effective even if removed from the development location.
- **Compliance Management:** The success of the protections can be continually and independently confirmed.
- **Code Testing and Verification:** Methods for code inspection are applied and suspicious code is detected.

To be effective in today's complex global supply chains, software integrity processes and controls must be designed to be independent of geography, accommodate diverse sources of software components, and extend from a vendor's suppliers to its customers.

Next Steps

The complexities and interdependencies of the IT ecosystem require software suppliers to not only be able to demonstrate the security of products they produce, but also evaluate the integrity of products they acquire and use. For this reason, every software supplier has a significant stake in the identification, communication and evaluation of best practices for ensuring software integrity. The challenge is to create practical but effective methods that build on a broad understanding of the dependencies and threats along the complex software supply chain. Ultimately this should lead to greater confidence through integrity checks incorporated in a defined secure development lifecycle.

While individual software companies have integrity assurance programs in place, there has been little industry-led effort to identify and share best practices for implementing the integrity controls described in this framework or to provide customers with more clarity into how the industry is addressing this issue. This is a critical gap that must be addressed by the software industry in order to continue to meet customer demands for both security and innovation.

To meet this important industry need, SAFECode will build upon this framework for software supply chain integrity with a focused effort to identify and analyze the most effective software integrity practices that its member companies use to help assure the integrity of their software. We will publish our findings later this year to extend these practices across the industry and provide customers with additional insight into how to view and evaluate the processes by which software integrity is achieved.

Though the focus of this immediate work is on the responsibilities of software suppliers, it should be noted that systems integrators and customers also play roles in the assurance of software in IT solutions. The proper execution of activities such as integration, configuration and implementation are crucial to the integrity of an overall IT system. For this reason, SAFECode will work with suppliers, integrators and customers to adopt a shared framework that extends software integrity across "systems of systems."

About SAFECode

The Software Assurance Forum for Excellence in Code (SAFECode) is a non-profit organization exclusively dedicated to increasing trust in information and communications technology products and services through the advancement of effective software assurance methods. SAFECode is a global, industry-led effort to identify and promote best practices for developing and delivering more secure and reliable software, hardware and services. Its members include Adobe Systems Incorporated, EMC Corporation, Juniper Networks, Inc., Microsoft Corp., Nokia, SAP AG and Symantec Corp. For more information, please visit www.safecode.org.

SAFECode	(p) 703.812.9199
2101 Wilson Boulevard	(f) 703.812.9350
Suite 1000	(email) stacy@safecode.org
Arlington, VA 22201	www.safecode.org

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity