



An Overview of Software Integrity Controls

A Software Assurance Approach to Minimizing Risks in the Software Supply Chain

Dan Reddy, CISSP, CSSLP

EMC, (SAFECode member company) Product Security Office
Co-chair of Supply Chain Whitepaper #2

Stacy Simpson, SAFECode Policy Director
Editor of Supply Chain Whitepapers

September 28, 2010

- The Software Assurance Forum for Excellence in Code (SAFECode) was announced on 23rd October 2007
- SAFECode is the first global, industry-led effort to identify and promote best practices for developing and delivering more secure and reliable software, hardware and services

10001
01111
10001
11110
10001

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

SAFECode Brings Together SMEs

- SAFECode unites subject matter experts with unparalleled experience in managing complex global processes for software development, integrity controls and supply chain security.
- The goal is not to establish one way but to identify methods that work and can be effectively leveraged in a rational way by governments and enterprises.

3

10001
01111
10001
11110
10001

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

SAFECode Objectives

- Increase understanding of the secure development methods and integrity controls used by vendors
- Promote proven software assurance practices among vendors and customers to foster a more trusted ecosystem
- Identify opportunities to leverage vendor software assurance practices to better manage enterprise risks
- Foster essential university curriculum changes needed to support the cyber ecosystem
- Catalyze action on key research and development initiatives in the area of software assurance

4



- Software could contain vulnerabilities whether inserted maliciously or not
 - Result of good coding practices preserved (secure development)
 - Guard against inadvertent vulnerabilities throughout lifecycle
 - Protect against malware introduced along the chain
 - Minimize possibility of intentionally inserted malicious code
- Attacks on easily found vulnerabilities likely to be more prevalent - given greater opportunity
- Can't defend against all threats. Minimize risk
 - Don't focus on the where code is developed
 - Focus on the practices to develop & maintain integrity



Software Assurance: Confidence that software, hardware and services are free from intentional and unintentional vulnerabilities and that the software functions as intended.

In practice, software vendors take action in three key, overlapping areas to achieve software assurance—security, authenticity and integrity.



Security: Security threats to the software are anticipated and addressed during the software's design, development and testing through secure engineering practices. This requires a focus on code quality and functional requirements to reduce unintentional vulnerabilities in the code.



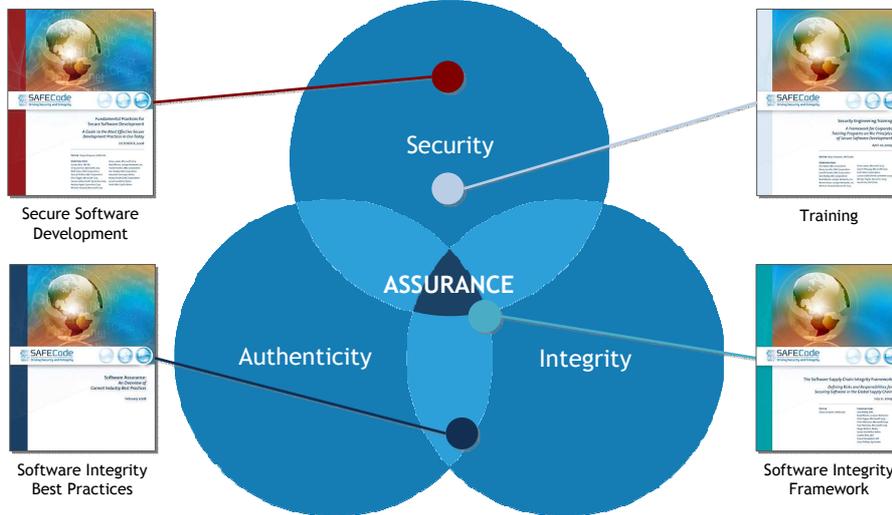


Integrity: Security threats to the software are addressed in the processes used to source software components, create software components, and deliver software to customers. These processes contain controls to enhance confidence that the software was not modified without the consent of the supplier.



Authenticity: The software is not counterfeit and the software supplier provides customers ways to differentiate genuine from counterfeit software.





Every software vendor manages three key lifecycle processes in their link of the software supply chain:

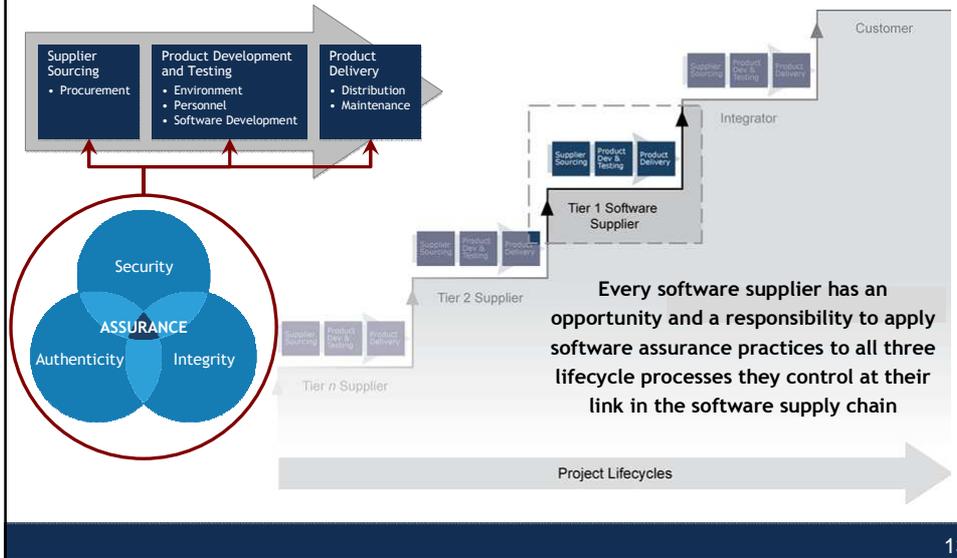
1. Software Sourcing: Vendors select their component and services suppliers, establish specifications for a supplier's deliverables and "on-board" software and hardware components and services received from suppliers.
2. Software Development: Vendors build, test, assemble, integrate and package components for delivery.
1. Software Product Delivery and Sustainment: Vendors deliver the software product to customers and provide ongoing maintenance.



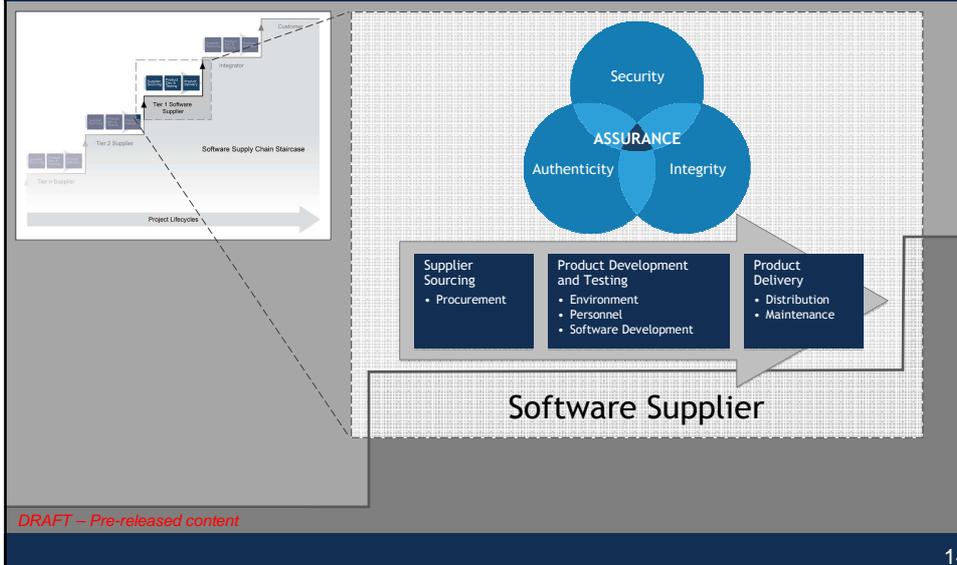
It is within each of these three processes that a vendor is able to implement effective software assurance practices to increase the security of the software they deliver.



Software Assurance In the Software Supply Chain



SAFECode's Focus

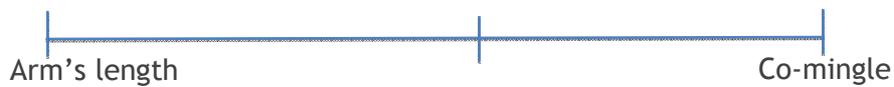




- Chain of Custody
- Least Privilege Access
- Separation of Duties
- Tamper Resistance and Evidence
- Persistent Protection
- Compliance Management
- Code Testing and Verification



- Nature of engagements vary



Supplier assures sound delivery

- Requirements vary, approach to validation of integrity varies

Database shipping with product
Control is indirect

Work for hire web site
Customer and Supplier
Teams collaborate,
share resources
More control



- Define Expectations of Suppliers in Agreements
 - Commitments to security testing
 - Code fixes
 - Vulnerability response
 - Stand behind Integrity & Authenticity Claims
 - Security training of teams
 - Warranty implications
- Follow industry specifications or practices where they exist and add value
 - E.g. Common Vulnerabilities and Exposures (CVE)
 - Reference development practices like SAFECode's paper or equivalent



- Open Source implementations vary
 - Some fully community based
 - Some supported by commercial entities
 - Depends on the relationship
- Shipped with COTS products sometimes
- Sometimes recommended or supported but not shipped
- Vulnerability response track record
- Vetting distribution sites
- Assess reputation of release engineering practices of vendor or community



- Secure Transfer
 - Authenticated endpoints and encrypted sessions
- Sharing of System & Network Resources
 - Leveraging digital identities for resource control
 - Map only needed resources for each project
 - Proper revocation of identities across suppliers
- Malware scanning at points of exchange
- Secure storage near transit points with access control
- Code exchange digitally signed



- People Security
 - Focus on organization and process controls
 - Background checks when appropriate - not a panacea
 - Caveats for global consistency and practicality
 - Separation of duties applies
 - Defense in depth means clear responsibilities for guarding code
 - Training
- Physical Security
 - Badge access, repository protection etc.
- Network Security
 - Risk based process for code assets during each project
 - E.g. IDS, source code protection on workstation & virtual machines
 - "Need to Know" access control
 - Disable, don't delete accounts upon termination



- Code Repository Security
 - Avoid risk of inadvertent errors too
 - Housing for all assets
 - Code manifest
 - Hardened servers
 - Secure configurations
 - Security Content Automation Protocol (SCAP) applicability
 - Tied to corporate authentication stores
 - Account privileges get management approval
 - Code segment access based on "Need to Know"
 - Detailed log entries with protection and retention
 - Develop source code handling policy & standard



- Build Environment Security
 - As automated as possible
 - Clear ownership
 - Scripts are assets too
 - Service accounts require accountability
- Peer Reviews
 - Scalability question but depends on how implemented
 - Should leverage testing for additional focus
- Security Testing (security and integrity work together)
 - Static Code Analysis Tools
 - Network and Web Application Vulnerability Scanners
 - Binary Code Analysis Tools
 - Malware Detection Tools
 - Security Compliance Validation Tools
 - Code Coverage Tools
 - Customer Verification Testing



- Dissemination
 - Malware scanning using multiple tools
 - Code signing
- Publishing
 - Secured processes. Available signatures and checksums
- Electronic transfers
 - Must be confirmable
- Authenticity
 - Cryptographically hashed or digitally signed components
 - Notification technology
 - Automatic verification during program execution



- Secure deployment
 - Secure by default & configuration options to customers
- Custom code extensions
 - Key implications for System Integrators
 - Must manage all assets and new code during integration and assembly since they are developing a solution



- Broader data needed
- Impact of Software and Cloud Computing
- More collaboration with traditional Supply Chain Management
- More measurement implications through Supply Chain
- Authenticity checks at run-time
- Authenticity verification - ease of use
- Testing that improves detection of malicious insertion
- Integrity is a key pillar of software assurance
- Software vendor is one link in complex chain
- Adoption of practices can increase confidence
- Encourage more refinement of practices