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

Software Assurance: *An Overview of Current Industry Best Practices*

February 2008

Executive Summary

Software Assurance: An Overview of Current Industry Best Practices

Software underpins the information infrastructure that governments, critical infrastructure providers and businesses worldwide depend upon for daily operations and business processes. These organizations widely and increasingly use commercial off-the-shelf software ("COTS") to automate processes with information technology. At the same time, cyber attacks are becoming more stealthy and sophisticated, creating a complex and dynamic risk environment for IT-based operations that users are working to better understand and manage. As such, users have become increasingly concerned about the integrity, security and reliability of commercial software.

To address these concerns and meet customer requirements, vendors have undertaken significant efforts to reduce vulnerabilities, improve resistance to attack and protect the integrity of the products they sell. These efforts are often referred to as "software assurance." Software assurance is especially important for organizations critical to public safety and economic and national security. These users require a high level of confidence that commercial software is as secure as possible, something only achieved when software is created using best practices for secure software development.

This white paper provides an overview of how SAFECode members approach software assurance, and how the use of best practices for software development helps to provide stronger controls and integrity for commercial applications.

Table of Contents

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

The Challenge of Software Assurance and Security

Software assurance encompasses the development and implementation of methods and processes for ensuring that software functions as intended while mitigating the risks of vulnerabilities, malicious code or defects that could bring harm to the end user. Software assurance is vital to ensuring the security of critical information technology resources. Information and communications technology vendors have a responsibility to address assurance through every stage of application development.

This paper will focus on the software assurance responsibilities of software vendors. However, integrators, operators and end users share some responsibility for ensuring the security of critical information systems. Because of the rapidly changing nature of the threat environment, even an application with a high level of quality assurance will not be impervious from attack if improperly configured and maintained. Managing the threats we face today in cyberspace requires a layered system of security, with vendors building more secure software, integrators ensuring that the software is installed correctly, operators maintaining the system properly, and end users using the products in a safe and secure manner.

SAFECode Software Assurance Definition:

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

New Risks and Countermeasures

The dynamic threat environment creates challenges for all software-related operations. Vectors for attacks that could interrupt or stop critical software functions must be considered in design and development. The software assurance risks faced by users today can be categorized in three areas:

1. **Accidental design or implementation errors** that lead to exploitable code vulnerabilities
2. **The changing technological environment,** which exposes new vulnerabilities and provides adversaries with new tools to exploit them
3. **Malicious insiders** who seek to do harm to users or vendors

Accidental Design or Implementation Errors

The prevalence of hackers, viruses, worms and other malicious software that attack systems and networks highlights the first risk area when programmers inadvertently create faulty software design or implementations. Developers address this risk through developer training and the use of secure development practices and tools. These processes are discussed in depth in the next section of this paper.

The Changing Technological Environment

Rapid change and innovation are two of the most enduring characteristics of the IT industry. But innovation is not unique to vendors. Criminals can and do innovate. In the span of only a few years a complex and lucrative criminal economy capable of supporting specialized skill sets for identifying and attacking software has developed.

The development of this sophisticated criminal economy contributes to increasingly targeted and complex attacks. Vendors commit resources to understand emerging threats and use state-of-the-art technologies, tools and techniques to develop software, hardware and services that can resist attack. The process is one of on-going improvement as new vulnerabilities are exposed, new threats are created and new countermeasures developed and implemented.

Malicious Insiders

There is a growing concern that global software development processes could be exploited by a rogue programmer or an organized group of programmers that would compromise software, hardware or services during the development process. Vendors are extremely protective of their "soft assets" such as their code base. The complex development process and the series of controls used to protect the development process provide powerful management, policy and technical controls that reduce these risks. There is no single way to manage or control a development process. Rather there are proven best practices that companies use to manage their unique development infrastructure and business models.

SAFECode members implement processes for vetting employees and contractors regardless of their country of residence. However, far more critical to software assurance is establishing and implementing processes and controls for checking and verifying software assurance irrespective of where it was produced.

From a development perspective, these controls are focused more on "how it was made" than "where they were sitting" during the coding process.

CASE STUDY

EMC Corporation

A centralized Product Security Office coordinates interrelated programs for strong security assurance at EMC Corporation.

Foundation: Product Security Policy Guides product development teams and is a common reference for product organizations to benchmark product security against market expectations and industry best practices. Metrics score company-wide use of the policy.

Knowledge: Security Training Role-based security engineering curriculum trains new and existing engineers on job-specific security best practices and how to use relevant resources.

Process: Security Development Lifecycle Overlays security on standard development processes for achieving a high degree of compliance with the above referenced Product Security Policy.

Architecture: Common Security Platform A set of software, standards, specifications and designs for common software security elements such as authentication, authorization, audit and accountability, cryptography and key management using state-of-the art RSA technology. An open interface allows integration with customers' security architectures.

Incident Response: Product Security Response Center Defines and enforces EMC's vulnerability response policy to minimize risk of exposure to customers.

External Validation: Security Certification EMC has received extensive government and industry certifications in design, implementation and management of its security processes and solutions – including Common Criteria or FIPS 140-2.

EMC² where information lives®

Managing Risk Through Software Assurance Best Practices

These risks can all be managed through the adoption of best practices in software assurance. While a number of international standards and certification regimes for software assurance have been issued, their effectiveness in achieving real-world reduction in vulnerabilities is debatable. Companies on their own have been taking the lead in developing and implementing practices to produce secure code that are better tuned to real-world software development processes and result in higher levels of security. SAFECode's mission, in part, is to bring these practices together to share across the community.

Industry Best Practices for Software Assurance and Security

Software vendors have both a responsibility and business incentive to ensure product assurance and security. Customers demand that software be secure and reliable. Vendors also must produce quality products to protect and enhance brand names and company reputations. These pressures motivate vendors to minimize mistakes in coding, reduce the occurrences of post-sale vulnerabilities and related patching, and to protect sensitive data and the operational integrity of customer IT systems.

To understand how vendors are earning the trust of customers, it is useful to examine best practices employed by the software industry and how they contribute to enhancing product assurance and security.

Software development processes vary by vendor according to their unique product lines, organizational structures and customer requirements. Not surprisingly, there is no single method for driving security and integrity into and across the globally distributed processes that yield technology products and services. Yet regardless of the method used, there is a core set of best practices for software assurance and security that apply to diverse development environments.

CASE STUDY
SYMANTEC CORPORATION

Symantec's product security framework, called Product Security Life Cycle (PSLC) shapes and governs the lifespan of products. It has nine steps: engagement and preparation, education and training, security goals and planning, risk assessment, adoption of best practices, building automated routine verifications, security testing, security readiness review and security response.

Implementation of the PSLC includes a series of extensive training classes about security awareness, secure development and security testing for members of the development and quality assurance teams. This knowledge is applied with state-of-the-art tools for effective and secure source code configuration management, product build, source code analysis, product test and defect remediation. Engineers routinely compile and check code modules and the entire system. Security testing is performed by quality assurance teams and a product security team.

Third-party components and open source software used in this company's products are subjected to additional requirements:

- Teams check all code for vulnerabilities using standard methodologies and tools;
- Providers are required to allow access to source code and/or that its vendor scan the code for common vulnerabilities;
- Teams have a documented, contractual service level agreement for security patches;
- Third-party code is implemented in a way that facilitates independent patching.

These efforts have earned leadership for this vendor in the certifications community. Many of its products are certified by Common Criteria, FIPS 140-2, ICSA Labs and Checkmark; manufacturing and distribution sites have ISO 9001 certifications.

Framework for Software Development

While there are several different development methodologies, they all share the following common elements:

Concept The initial phase of every software development lifecycle is to define what the software is supposed to do, how users will interact with the product, and how it will relate to other products within the IT infrastructure. This is when product development managers assemble the team to develop the product.

Requirements This phase translates the conceptual aspect of a product into a set of measurable, observable and testable requirements. Developers phrase these requirements as "the product shall…" and specify exactly what functions will be provided, including related degrees of reliability, availability, maintainability and interoperability. It is crucial for the requirements phase to explicitly define functionality as this will affect subsequent programming, testing and management resources expended in the development process.

Design and Documentation Efficient programming requires systematic specifications of each requirement for a software application. This phase is more than an explicit, detailed description of product functionality. The level of detail in this phase will adequately enable production of near-final drafts of documentation to coincide with final release of the product.

Programming This phase is where programmers translate the design and specification into actual code. Effective coding requires implementers to enforce consistent coding practices and standards throughout all aspects of producing the application. Best practices for coding ensure that all programmers will implement similar functions in a similar manner. Programmers require appropriate training to ensure implementation of these standards.

Testing, Integration and Internal Evaluation This function verifies and validates coding at each stage of the development process. It ensures that the concept is complete, that requirements are well-specified, measurable, and that test plans and documentation are comprehensive and consistently applied to all modules, subsystems, and integrated with the final product. Verification and validation occurs at each stage of development to ensure consistency of the application. Complex projects require testing and validation methodologies that anticipate potentially far-fetched circumstances**.** That testing simulates the kind of duress that an attacker might apply to break an application.

Release This phase makes the application available for general use by customers. Before releasing the application, a software provider must ensure that the application meets product criteria, identify delivery channels, train the sales organization to match target buyers with the product's functionality, and fulfill orders. The application's vendor support team must be able to respond to customer queries at production volumes worldwide.

Maintenance, Sustaining Engineering and Incident Response These processes support released products. Applications must be updated with bug fixes, user interface enhancements, or other modifications meant to improve the usability and performance of the product. Defects fixed in this phase of the product lifecycle are merged into the subsequent version of code, and analysis is conducted to mitigate the possibility of their recurrence in future versions or other products.

CASE STUDY

Juniper Networks

Juniper Networks implements a TL 9000 certified process for managing product development. This process is audited regularly and protects the integrity of our products while providing accountability and predictability.

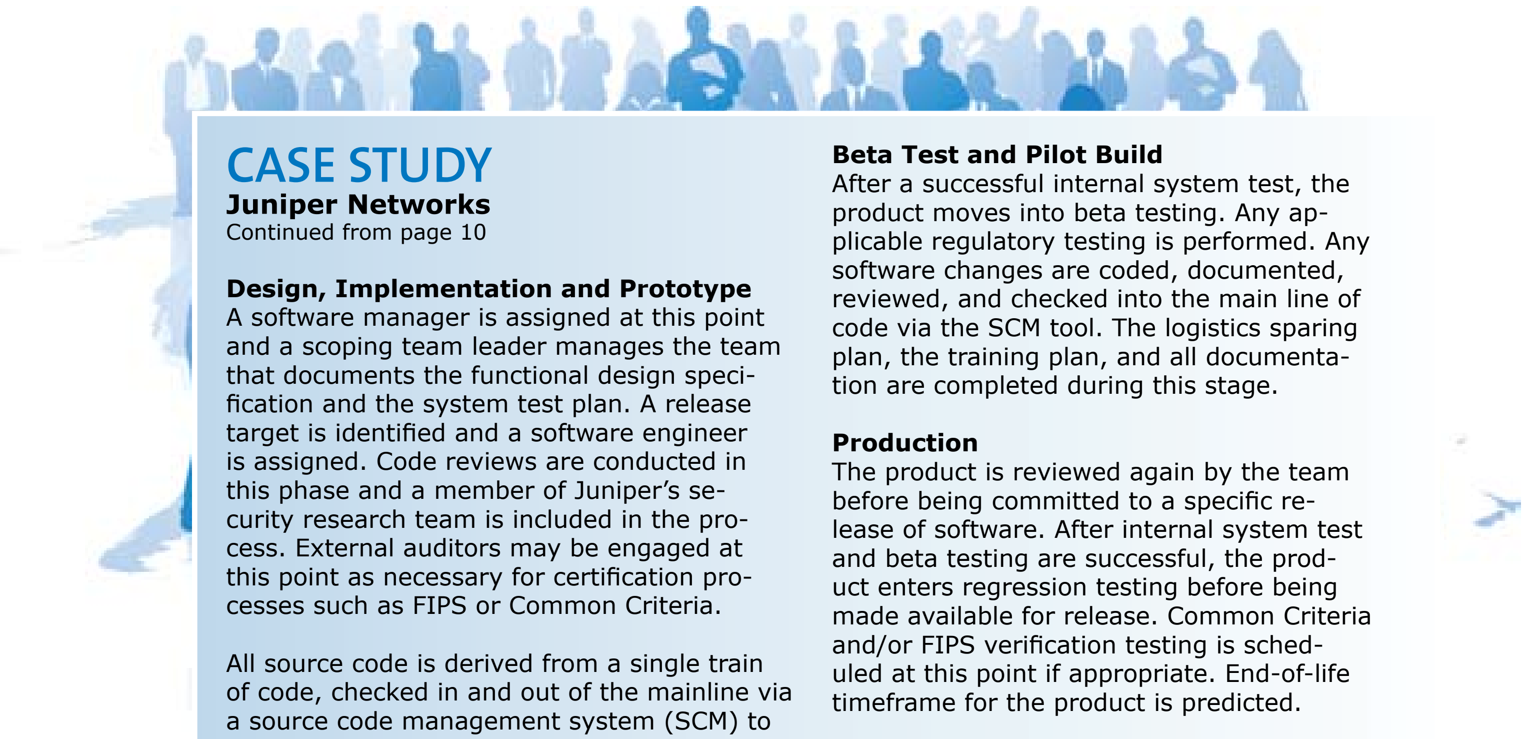
Projects are managed from concept to end of life (EOL) via a 7 phase process that includes:

- Concept and Feasibility
- Plan and Specifications
- Design, Implementation and Prototype
- System Test
- Beta Test and Pilot Build
- Production
- End of Life

Concept and Feasibility
Requirements are defined, tracked, and managed via a database in this phase of the process. If the requirement originated from a customer, then the customer must approve the requirements document to ensure the design is what the customer really wanted.

Plan and Specifications
The development and delivery schedule is identified in this phase. The engineering team is defined, including software engineering, a scoping team leader, and a product manager. A manufacturing plan, as well as a diagnostic test plan is defined in this phase.

Continued....



CASE STUDY

Juniper Networks

Continued from page 10

Design, Implementation and Prototype
A software manager is assigned at this point and a scoping team leader manages the team that documents the functional design specification and the system test plan. A release target is identified and a software engineer is assigned. Code reviews are conducted in this phase and a member of Juniper's security research team is included in the process. External auditors may be engaged at this point as necessary for certification processes such as FIPS or Common Criteria.

All source code is derived from a single train of code, checked in and out of the mainline via a source code management system (SCM) to track changes, and any changes made must be documented and peer reviewed. Juniper utilizes a company-wide bug tracking system that is integrated with our SCM, and all bugs are assigned a bug tracking number.

System Test
Hardware and software unit testing is performed and the reports are reviewed in this phase. Products are evaluated by an internal team made up of system test, software engineering, the software manager, hardware engineering, and technical publications. Code reviews and code scanning tools are employed to minimize mistakes and vulnerabilities. If penetration testing is appropriate it is conducted at this point. Beta plans are defined, and training plans are then created. Prototypes are built during this phase.

Beta Test and Pilot Build
After a successful internal system test, the product moves into beta testing. Any applicable regulatory testing is performed. Any software changes are coded, documented, reviewed, and checked into the main line of code via the SCM tool. The logistics sparing plan, the training plan, and all documentation are completed during this stage.

Production
The product is reviewed again by the team before being committed to a specific release of software. After internal system test and beta testing are successful, the product enters regression testing before being made available for release. Common Criteria and/or FIPS verification testing is scheduled at this point if appropriate. End-of-life timeframe for the product is predicted.

Product bugs or vulnerabilities during production are reported, assigned a number, and tracked in a bug tracking system. Resulting code changes are tracked within SCM, the same as initial code design. The code changes are again peer reviewed and code scanning may be employed if appropriate. After verification of code, the product is regression tested and scheduled into a maintenance release.

End of Life
The product end of life is formally announced and the notice is posted to our customer service website.

Juniper NETWORKS

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

Software Security Best Practices

In each stage of the software development lifecycle defined above, there are best practices for instilling security in a software application. Across SAFECode's membership, the following security best practices and controls are well established:

Security Training A prerequisite to developing secure software is for the development team to be well-versed in information security – including security and privacy issues that may affect people who use the product. Some vendors use external trainers to deliver security training to their product developers. Other vendors have established in-house trainers and online educational content to customize the training to their specific technologies and applications. Training topics include a wide range of issues such as how to do threat modeling, role-based security engineering, avoiding unsafe library function calls and preventing cross-site scripting errors. Trainers leverage the available published materials from industry and academia.

Defining Security Requirements Security requirements must be defined during the early stages of product development, especially the requirements definition stage. Security requirements must go in tandem with product development and therefore address architecture and design, product development and programming best practices, and requirements for assurance, testing and serviceability. Security requirements set at the outset of a product development cycle may include specific security metrics and goals for each major phase of development. Some teams measure the effectiveness of design security reviews or code audits as well as security testing goals. These requirements are set at the beginning of the project and then checked during the development cycle. Quality Assurance teams will set their security testing goals during this phase.

Secure Design The early design phase must identify and address potential threats to the application and ways to reduce the associated risks to a negligible level. These objectives may be accomplished with threat modeling and mitigation planning, which includes analyzing the system, and potential vulnerabilities and attack vectors from an adversary's perspective. Some vendors formalize their attack vector analysis through threat modeling. Security experts can be brought in to help facilitate this process of identifying potential threats and developing designs that mitigate those threats.

Secure Coding The product development team must implement secure programming practices. This is where programmers exercise the secure coding skills they learned during their training. These require inspection of an application's source code to identify vulnerabilities induced by coding errors, and implement secure programming practices that reduce the frequency and severity of those errors. Examples of secure coding practices include source code review using a combination of manual analysis and/or automated analysis tools for identifying potential security defects.

Secure Source Code Handling Security best practices include careful handling of source code, including tight change management and tracking and confidentiality protection of code such that only authorized persons are permitted to view or modify its contents in order to prevent malicious insiders from introducing vulnerabilities. Systems that process or handle source code must be protected from unauthorized access inside or outside the developing company, and from intentional or unintentional unauthorized modification. Design and code reviews are also conducted as a way of preventing malicious insertion of code.

Security Testing Security testing is specialized validation that ensures that the security requirements were met and the secure design and coding guidelines were followed. Testing may include vulnerability analysis, penetration testing, or use of security testing techniques such as "fuzzing" or varying external inputs to identify potential buffer overflows and other errors. Some vendors not only do internal testing, but also submit their products to external testing or certification. Penetration testing by independent teams can uncover vulnerabilities that would not be detectable using other means.

Security Documentation Software product documentation must include explicit treatment of security issues to help customers understand how to optimally configure security controls, and how configuration options may or may not expose potential security vulnerabilities. Examples include creating a Security Configuration Guide as a standard part of product documentation.

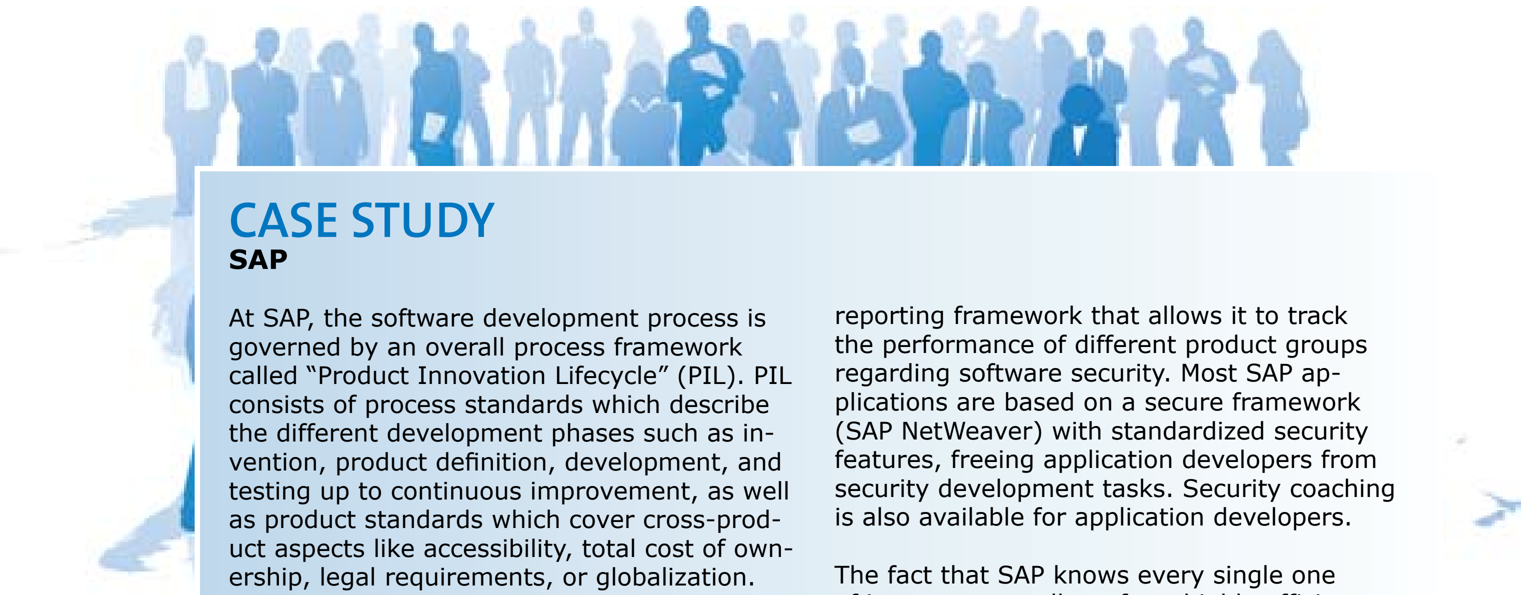
Security Readiness Just before releasing a product, the application developer must evaluate, document and assess risks posed by potential security gaps in the product. This risk management best practice enables a product development organization to evaluate the security posture of a product and whether it is safe to proceed with its release to general availability. For some vendors, this phase is where a final check is done to ensure that all of the security requirements set in the requirements phase have been met.

Security Response Any security vulnerabilities (exploited or not) reported against the deployed product are handled through incident response and relayed to the product development or sustaining teams to mitigate the vulnerability. Communication with the discoverers and the customers is important to ensure that proper actions are taken to mitigate the risk. In some cases, this may mean the vendor will issue a patch to the product. Some vendors have developed technologies that enable customers to receive security patches automatically to minimize their exposure to risks.

Integrity Verification Some products offer customers methods such as signed code for verifying that the software they have acquired is indeed from their trusted vendor. Using public key technology to sign code is an example of enabling integrity verification. Some software companies also build in integrity checks on an on-going basis to assure that the components in the solution are indeed bona-fide components.

Security Research Developers learn to adapt new technologies to provide greater customer capability and value. Along with this investigation comes research into new threat vectors and mechanisms to mitigate them. Similarly, as new attack vectors against existing technologies become known, developers implement mechanisms to defend against them.

Security Evangelism Leaders in the area of software assurance promote the use of best practices by discussing their practices and findings in open forums, articles, papers and books. SAFECode is a central forum for promoting the use of best practices to those who need guidance.



CASE STUDY

SAP

At SAP, the software development process is governed by an overall process framework called “Product Innovation Lifecycle” (PIL). PIL consists of process standards which describe the different development phases such as invention, product definition, development, and testing up to continuous improvement, as well as product standards which cover cross-product aspects like accessibility, total cost of ownership, legal requirements, or globalization.

Security is a product standard within PIL. The standard has evolved from a number of sources, including SAP development experience, know-how contributed by renowned security specialists, market trends, customer feedback, legal requirements, SAP strategy, and research findings. It consists of a security planning framework with best practices for addressing common security issues, and a security report that reflects the status of the implementations defined in the security plan after development. In addition, it includes test case descriptions for a number of requirements.

The security standard represents the core of secure software delivery at SAP. It is complemented by security training for developers and testers, in-house security tests, white-box and black-box security hacking by external partners on selected top-priority components, as well as a global product security reporting framework that allows it to track the performance of different product groups regarding software security. Most SAP applications are based on a secure framework (SAP NetWeaver) with standardized security features, freeing application developers from security development tasks. Security coaching is also available for application developers.

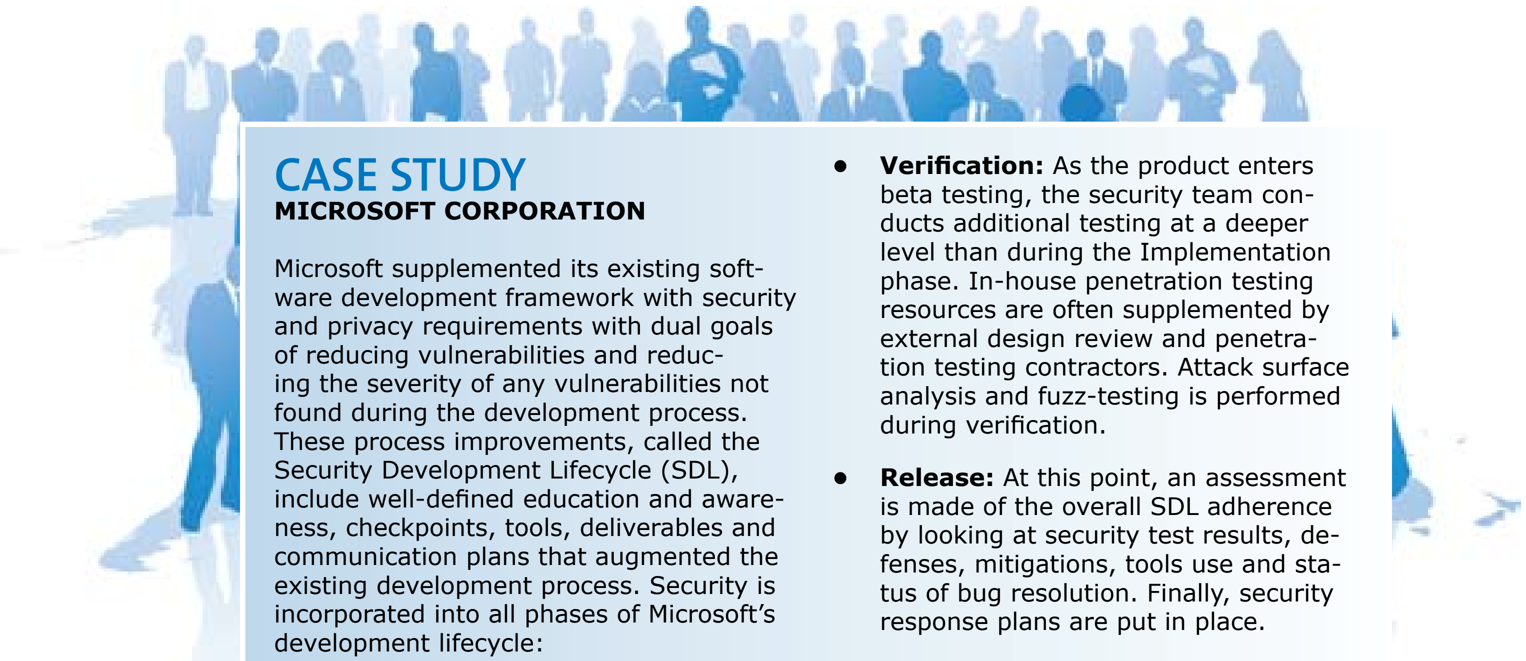
The fact that SAP knows every single one of its customers allows for a highly efficient security management. Security issues can be communicated privately to customers via “Hot News”, eliminating the need for public announcements. However, the research community has been showing a growing interest in SAP software. To provide first-hand information and create transparency, SAP maintains security forums and publishes newsletters. Customers and researchers can also contact the SAP Security Response Team directly via security@sap.com.

In addition, the SAP Security Optimization Service can be used to check a customer’s security status, and a staff of highly-qualified security consultants is available for remote or on-site support in security questions.

Related Roles of Integrators and End Users

The best practices described above are aimed squarely at software vendors and their global supply chain of developers. Software assurance, however, does not end with the vendor. It is a continuous process. The broader ecosystem of software integrators, operators and end users who buy and deploy the applications all contribute to the overall assurance of a product or a system.

- **Integrators:** As applications are scaled to very large environments and integrated with other products and legacy systems, new vulnerabilities that did not exist in the stand-alone product may be introduced. Integrators must work in partnership with software vendors to find and mitigate these vulnerabilities.
- **Operators:** Operators must ensure that systems remain properly configured. Automated patching should be enabled to speed the remediation of vulnerabilities. Operators must also deploy standard layered defense measures for security, such as firewalls, antivirus, anti-malware, anti-phishing, intrusion detection and prevention, virtual private networks, strong authentication and identity management.
- **End users:** End users must take the responsibility to report potential bugs or vulnerabilities and must not introduce software from untrusted sources into systems. Responsible use of software is an important ongoing requirement for assurance and security.



CASE STUDY

MICROSOFT CORPORATION

Microsoft supplemented its existing software development framework with security and privacy requirements with dual goals of reducing vulnerabilities and reducing the severity of any vulnerabilities not found during the development process. These process improvements, called the Security Development Lifecycle (SDL), include well-defined education and awareness, checkpoints, tools, deliverables and communication plans that augmented the existing development process. Security is incorporated into all phases of Microsoft's development lifecycle:

- **Requirements:** Determine security certification requirements and list of processes and tools that must be used in development process.
- **Design:** The product team defines the product's security architecture and design guidelines, calculates how much of the product is exposed to untrusted users (called the attack surface), conducts threat modeling to uncover "at-risk" components, and defines criteria for shipment to minimize vulnerabilities.
- **Implementation:** The product team creates the product and threats are mitigated through updated libraries, secure coding standards, security testing and use of code analysis tools, and many defense-in-depth methods must also be employed.
- **Verification:** As the product enters beta testing, the security team conducts additional testing at a deeper level than during the Implementation phase. In-house penetration testing resources are often supplemented by external design review and penetration testing contractors. Attack surface analysis and fuzz-testing is performed during verification.
- **Release:** At this point, an assessment is made of the overall SDL adherence by looking at security test results, defenses, mitigations, tools use and status of bug resolution. Finally, security response plans are put in place.
- **Support and Servicing:** A central security response team handles externally reported vulnerabilities. The central security team works closely with the security response team and uses information about newly reported vulnerabilities to update tools, education materials, coding standards, and potentially the security development process to minimize vulnerabilities in future product versions.

Microsoft is active in the certifications community and regularly pursues FIPS 140 and Common Criteria evaluations of various software products and components. Microsoft addresses feature requirements for certification during the Requirements phase of development.

Microsoft®

SAFECode's Goals

SAFECode will drive stronger software security and integrity across the IT ecosystem by producing a series of products that build on this first white paper. Each product will tangibly advance software security and integrity over the next five years. The papers will address five key goals:

A Comprehensive Knowledge Base: Developers are trained and educated in secure coding practices and certification programs are well developed. Customers and developers understand the importance of software assurance and realize the return on investment.

Strong Development Processes: Developers implement processes that are demonstrated to be effective at improving security and have a clear roadmap for beginning the process of building robust software assurance programs.

Consistent Evaluation, Metrics and Certification: The quality of code can be judged based on both the inputs (strength of the development processes and training and education of developers) and the outputs (reductions in downtime, vulnerabilities and new malicious code).

Effective Response Processes: Processes for identifying and remediating newly discovered vulnerabilities are routinized across the software ecosystem.

Rigorous R&D: The most important software assurance R&D needs are identified and supported with the appropriate resources.

Conclusion

The global software industry is making great strides at improving software assurance and security by applying best practices described in this white paper. Vendors who have implemented best practices are already seeing a dramatic improvement in software product assurance and security. These are practices that should be considered, tailored and adopted by every software developer and vendor. The result of efforts like these will be a higher level of confidence for end users in the quality and safety of software that underpins critical operations in governments, critical infrastructure and businesses worldwide.

Questions for Vendors about Product Assurance and Security

SAFECode invites your organization to study how vendors use these best practices as part of the product procurement process. The following are questions you might pose to determine the assurance and security of a proposed product procurement or vendor engagement.

- What are your best practices for software assurance?
- What are your best practices for software security?
- Who in your company is responsible for software assurance and security and how do they manage the processes?
- How are these best practices implemented by contractors and other members of your global supply chain?
- How does your company assure the quality and security of publicly available software modules and libraries used within your products?
- How does your company assure that implementation of standards-based protocols for networking functionality is robust and safe?
- How much has use of these best practices decreased defects and vulnerabilities in your software products?
- How does your procedure for patching facilitate non-disruptive operation of your software applications?

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 proven software assurance methods. Founded by EMC Corporation, Juniper Networks, Inc., Microsoft Corporation, SAP AG and Symantec Corp., SAFECode works to identify and promote best practices for developing and delivering more secure and reliable software, hardware and services. For more information, please visit www.safecode.org.

SAFECode
2101 Wilson Boulevard
Suite 1000
Arlington, VA 22201

(p) 703.812.9199
(f) 703.812.9350
(email) inquiries@safecode.org
www.safecode.org

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