



Security in Development: The IBM Secure Engineering Framework



Redguides
for Business Leaders

Danny Allan
Tim Hahn
Andras Szakal
Jim Whitmore
Axel Buecker



- Investigating common development processes and the IBM Integrated Product Development process
- Emphasizing security awareness and requirements in the software development process
- Discussing test and vulnerability assessments



Executive overview

IBM® has long been recognized as a leading provider of hardware, software, and services that are of the highest quality, reliability, function, and integrity. IBM products and services are used around the world by people and organizations with mission-critical demands for high performance, high stress tolerance, high availability, and high security.

As a testament to this long-standing attention at IBM, demonstration of this can be traced back to the Integrity Statement for IBM mainframe software, originally published in 1973:

IBM's long-term commitment to System Integrity is unique in the industry, and forms the basis of MVS (now z/OS®) industry leadership in system security. IBM MVS (now z/OS) is designed to help you protect your system, data, transactions, and applications from accidental or malicious modification. This is one of the many reasons IBM 360 (now System z®) remains the industry's premier data server for mission-critical workloads.

This commitment continues to apply to IBM's mainframe systems and is reiterated at this site: http://www.ibm.com/servers/eserver/zseries/zos/racf/zos_integrity_statement.html

The IT market has certainly transformed in 35-plus years, and so have product development and information security practices. The IBM commitment to continuously improving product security has remained a constant differentiator for the company.

In this IBM Redguide™ we describe *secure engineering practices for software products*. We offer a description of an end-to-end approach to product delivery, with security taken into account. IBM is publishing this in the hope that interested parties - whether they be clients, other IT companies, academics and others - can find these practices to be a useful example of the type of security practices that are increasingly a must-have for developing products and applications that run in the world's *digital infrastructure*. We also hope this IBM Redguide can enrich our continued collaboration with others in the industry, standards bodies, government, and elsewhere, as we seek to learn and continuously refine our approach.

Let us take a look at some background information.

IBM is involved in the following three core areas of software development:

1. Development of products and solutions for sale.
2. Development and operation of solutions and services for its own internal use.
3. Development and operation of solutions and services on behalf of clients.

To help meet client demand for flexible and full-function information technology solutions, IBM software product development teams design products that integrate with, and operate within, a wide range of operating systems and programming language environments. IBM products, solutions, and services may integrate IBM developed software, open source code, third party code, and potentially customized extensions or applications into composite products and solution offerings.

The development of IBM products and solutions is distributed across organizations and laboratories worldwide. The magnitude of the secure engineering and process control challenges involved in producing high-quality software in such a global development environment is significant.

The key to delivering products and services that meet clients' high expectations is to focus product development execution in four critical areas (Figure 1): a Common Development Process; a Secure Engineering Framework; a Continuous Security Improvement model; and a Supply Chain Security process. The combination of process, framework, and model integrate with a broader set of externally facing processes referred to as *global supply chain management*.

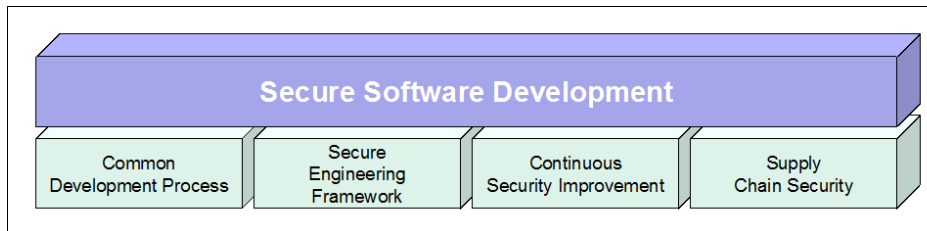


Figure 1 Software assurance at IBM

In the remainder of this Redguide we provide details about the elements of secure product development.

Common development process

Information Technology organizations should employ a *common development process* to provide consistent management, technical oversight, and accountability across a wide range of hardware, software, services, and solution development projects. To achieve high levels of efficiency, quality, and security, the common development process should be supported by a set of enforceable and measurable standards and directives.

In addition to providing for accountability and control, the common development process enables the coordination of people, technology, and information involved in the development lifecycle of components, products, and solutions.

Development projects should be outcome-oriented. Each team should have sufficient flexibility to adopt tools and practices that can enhance their ability to deliver, as long as the results meet the governance criteria and follow the overall development process.

An *Integrated Product Development (IPD)* process was created in IBM as a result of quality-driven process re-engineering that has occurred over many years. The IBM IPD process has been adapted to provide a structure for software, hardware, and services development projects. Its core concepts have remained intact as the industry and associated development tools and methodologies have evolved. Each project is guided through a series of project phases under the oversight of the Project Development Team, or PDT. Figure 2 shows the structure, the inputs, the outputs, and the phase activities.

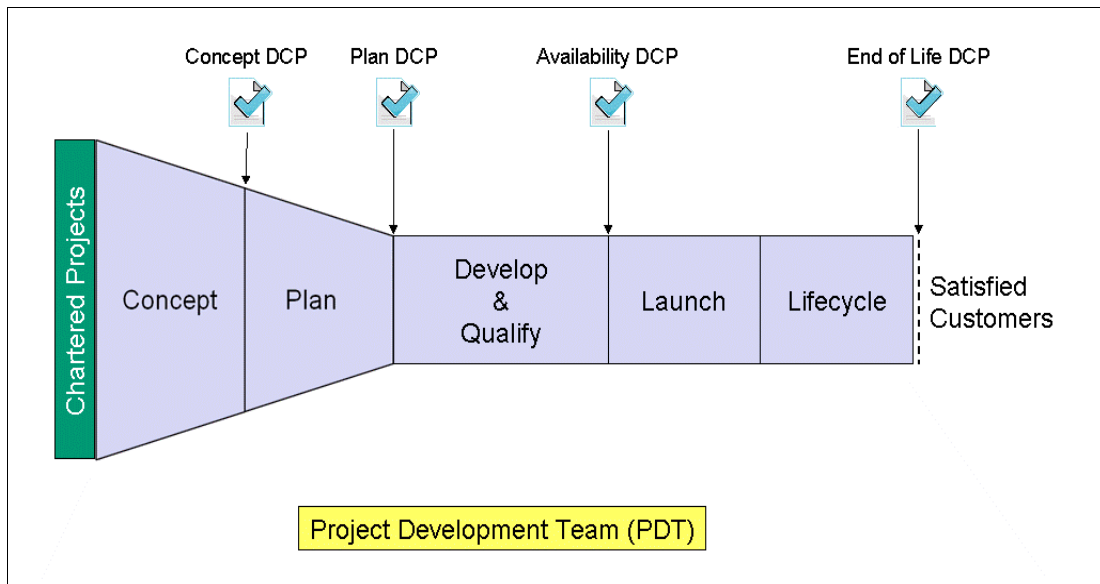


Figure 2 An Integrated Product Development process

This version of a common development process provides for a multi-phase project plan and sequenced activities and tasks for understanding customer requirements, conceptualizing, planning, developing, testing, delivering, and supporting the components, products, and solutions that respond to those requirements.

At the *business level* this process serves to enable informed risk and investment decisions. At the *operational level* it facilitates compliance with corporate directives and adherence to plans and checkpoints. At the *design, implementation, and delivery levels* this process helps ensure that peer design reviews are conducted; that test plans are executed; that information assets are protected; and that products are developed to acceptable quality tolerances.

Checkpoints before availability of the offering help ensure that appropriate testing has been completed and that the quality of the offering is acceptable.

Governance of a common development process

In support of the development process, organizations should establish governance in the form of standards, practices, and compliance criteria. Four important elements of governance for development of hardware, software, and services are:

- ▶ Protection of assets
- ▶ Development project check points
- ▶ Security and quality plans
- ▶ Product testing

Protection of assets

Because development projects are never completely isolated from the rest of an organization, it is important to ensure that governance is not limited in scope to development projects.

The IBM Business Conduct Guidelines define proprietary information to include *software in object or source code format*. Personnel complete an annual mandatory training on these Business Conduct Guidelines, covering *Intellectual Property Protection*, *Corporate Security Standards*, and *Export Regulations*. It is a condition of employment that every IBM employee must demonstrate an understanding of, and commit to compliance with, the directives, the standards, the processes, and the practices related to their roles in IBM. Protection of proprietary information is one of those responsibilities.

Product development checkpoints

As shown in the lower portion of Figure 2 on page 3, it is recommended that projects within the common development process be separated into phases, such as: *concept*, *plan*, *develop*, *qualify*, *launch*, and *lifecycle support*.

This structure provides an opportunity for the project development team to conduct development checkpoint reviews as the project transitions from one phase to the next. These checkpoints can be used as control points for assessing project risk, expense control, product quality, issue review, and for project plan synchronization.

In order for the project to move to the next phase, the project development team should be required to satisfy the success criteria for the prior work, as well as justify any deviations from the plan, such as a change in scope or content. The projects should be required to address open issues before proceeding to the next project phase.

Security and quality plans

Every development project within an organization should require both a *security plan* and a *quality engineering plan*. These plans detail the technical and audit requirements for asset control, along with the standards and practices for quality engineering to be applied in the development process.

As cited earlier, a key element in the security plan is protection of proprietary information. A team within the organization should be responsible for setting appropriate data classifications and for overseeing the protection of the organization's proprietary information assets within the development process.

Organizations should require that all product development projects prepare a quality plan that describes how the project will meet corporate standards. Prior to availability to customers, a

review of the product's quality results relative to the plan is performed to validate how the project has met these standards.

As a specialization of quality engineering, organizations should maintain a community of practice for secure software engineering. This secure engineering program should establish a measurement system of *continuous security improvement* as a fundamental part of a secure product development strategy. The secure engineering program should be executed in four parallel and intertwined pillars:

- ▶ A mandate for continuous security improvement in technology and manufacturing drives accountability and action.
- ▶ A community of software engineers that innovate and share practices and tools for secure product development.
- ▶ Integration across products that is achieved through client use cases, scenarios, and end-to-end usage threads in concert with an architectural framework that enables componentization.
- ▶ Consumability analysis that looks beyond product defects to the client experience of using the offerings.

All of the security and quality plans, practices, and findings should be reviewed in the development phase checkpoint meetings.

Product testing

All products should undergo a range of tests in order to verify functional operation in accordance with the official design specifications of component, product, or solution. This includes verification of the security mechanisms and services incorporated into a component, product, or solution.

IBM development teams perform several levels of testing during development projects, including:

1. *Unit Test* verifies that a software element, subroutine, or class performs as designed in isolation.
2. *Component or Function Verification Test* verifies that a composite software element operates in accordance with written specifications.
3. *System Verification Test* verifies the integration and operation of components and products within the full solution environment.

Security testing is performed during Component Verification Test and System Verification Test. Security testing might include automated testing using tools such as *IBM Rational® Software Analyzer* and *IBM Rational AppScan®* as well as security testing using ethical hacking techniques.

Where appropriate, products might undergo outside analysis and testing, including certification as specified by the Common Criteria¹.

The project development teams should review the results of unit testing, component testing, system testing, security testing, and certification testing during the phase checkpoint meetings. Defects should be returned to the change team within development for rework and subsequent retest and verification by test teams. The project development team leader should hold the authority to prevent the project from progressing from the Development Phase to the Availability Phase until the *test exit criteria* have been met.

¹ For more information visit <http://www.commoncriteriaportal.org>

Product life cycle management

Once a product is made available, attention to security should continue in the product support channel. An organization should establish internal processes that allow for the notification of clients regarding high pervasive (HIPER) fixes that are recommended to be applied.

For those issues that have security implications, organizations should coordinate with vendors, researchers, and other bodies such as the Industry Consortium for Advancement of Security on the Internet (ICASI)² and Computer Emergency Response Team (CERT)³ to investigate and responsibly disclose information and remediations for vulnerabilities.

Organizations should also have internal mechanisms to help ensure that managers of potentially affected products are notified quickly of security issues that might arise. This is required since products often share componentry.

Common development process summary

A development process defines the overall steps for developing software and solutions delivered to clients of an Information Technology organization. It can provide the structure for conducting development projects, and facilitating compliance with corporate standards and practices. Deploying a common development process fosters consistent management, technical oversight, and accountability across a wide range of hardware, software, services, and solution development projects.

Secure Engineering Framework (SEF)

In addition to a common development process, organizations should give particular attention to the security characteristics of the offerings they create. Organizations might find it useful to create a *Security Architecture Board* (SAB)⁴ to help maintain a set of recommended guidelines and best practices to guide development teams in building more secure software. The *Secure Engineering Framework* (SEF) is intended to help ensure that software is secure by design, secure in implementation, and secure in deployment. The global nature of software development activities today necessitates the application of secure engineering principles across global development teams regardless of their physical location.

The Secure Engineering Framework that we describe in this document includes sections on education and awareness, project planning, risk assessment and threat modeling, security requirements, secure coding, test and vulnerability assessment, documentation, and incident response. The following material has been selected from the framework as an example.

Education and awareness

Unfortunately, examining the current state of the technology industry reveals that many security exposures occur because development organizations are unaware of the root causes of these security vulnerabilities and their impact. Development teams frequently work under tight deadlines and are under pressure to squeeze in as many product features as possible. When the importance of software security is not understood, it can become an afterthought.

² For more information visit <http://www.icasl.org>

³ For more information visit <http://www.cert.org>

⁴ A Security Architecture Board can be a team of security experts gathered from across the organization that is responsible for coordinating security offerings, messages, and technical directions across the organization.

Developing secure hardware and software requires that product managers, development managers, system and software architects, developers and quality assurance specialists have the knowledge and skills to build secure systems and software.

Improving overall security requires that organizations implement security awareness programs for their development teams. Because development teams consists of members who have a variety of roles a security awareness program must be designed to provide the appropriate type of information to each audience. For example, product managers need to understand the impact of security issues on customers. However, they might not need to know the deep technical details about the causes of security issues and the programming techniques to resolve them. These are concepts that developers need to learn.

A typical set of roles in a system or software development organization and the type of security knowledge and skills each requires are considered in the following sections.

Product managers

Product managers should understand customer problems and translate these into product requirements. They should understand the level of security that is necessary and ensure that it is described in their requirements documents. Therefore, product managers can benefit from training on the fundamentals of system and software security and specifically the impact of security issues.

Product managers should be aware of the industry standards and government regulations related to system and software security. Product managers must also understand the customers' expected usage models.

Development managers

Development managers should understand the effect of security issues on the users of their applications. They need to have general knowledge of the technical nature of these issues. In addition, they should learn about secure development frameworks and methodologies, such as *threat modeling* and *risk assessment*, and learn how to implement these during the different phases of their product's software development process.

System and software architects

System and software architects benefit from training on the technical nature of security issues as well as training on secure coding principles and techniques. They should learn the security features and issues related to their development platform so that they can design solutions that meet the security requirements. System and software architects should understand *threat modeling* and *risk assessment* techniques so that they can be applied to their products. System and software architects should articulate the potential threats to their designs.

Developers

Developers are responsible for writing code (including microcode in systems) that is free of security vulnerabilities and free of viruses, malicious code, back doors/trap doors and other potential weaknesses. Developers are also responsible to help ensure that configuration and integration of components within a larger product or offering does not introduce, or facilitate, security vulnerabilities. This requires that they understand the coding mistakes that lead to security issues and the principles for secure coding, as well as being able to test their own code.

Quality assurance personnel

Testing specialists need to understand the security issues they must watch out for. They can also benefit from training on application security testing techniques and methodologies as well as training on different security testing tools.

Implementing a security awareness program for development

Security awareness is essential for all key stakeholders in the development team. There is an abundance of available information about software security—courses, books, articles, and so on. It is important that appropriate learning methods and resources are selected for the product development team. For example, it might not be practical to send the entire development team to a five day course on software security. It might be more appropriate to send some members of the team to a course like this and consider investing in Web-based training for the rest of the team. A learning plan could also include reading relevant books and articles.

An essential task of a *team leader* is to evaluate the security awareness of the team, gauge the level of success of your training program, and schedule appropriate additional education or refresher courses as necessary.

Project planning

Each development group should start planning for security right from the beginning to avoid expensive rework as a result of security vulnerabilities that are discovered late in the development cycle. It is generally accepted that fixing defects earlier in any development cycle is more cost-effective than finding and fixing defects later in the development cycle. This is true for security-related defects as well. Further, avoiding the loss of confidence by customers is an added incentive to find and fix security-related issues during the development of products.

Further, security analysis and testing should be integrated into each major phase of the product's development cycle regardless of the methodology that is being used.

Regardless of the software development process followed, be it *waterfall-based*, *iterative*, or *agile*, every product development team spends time evaluating requirements, designing, coding, testing, and maintaining. The time and scope might vary greatly between the methods used, but the basic phases still exist with associated security practices.

During project planning, the development team should account for security analysis, requirements, design, testing, and documentation work. A checklist of basic items in a development plan includes:

1. Are the right people, with the right skills, on or available to the development team to perform the security work?
2. Has a security risk assessment and architectural review been performed?
3. Are new security features needed or is it necessary to modify existing features?
4. Is there a test plan in place and are tools available to perform security testing?
5. Has the development team gathered the latest information about security threats and vulnerabilities in the technology and the target operating environments for the component, product, or solution?
6. Has adequate time been factored into the schedule for security testing and fixing any security vulnerability issues found?
7. Is there a documentation plan that includes sections related to security and securing the offering?

Organizations should be sure to require that applications with higher risk of exposure make an increased investment in creating a security plan for the project.

Risk assessment and threat modeling

Projects should begin with a very simple *Predictive Threat Index* (PTI) calculation. This simple and prescriptive calculation allows a team to document the relative business value of information and processing handled by the software or solution. This value is used to estimate the level of effort required to achieve the desired security characteristics of the software.

Threat modeling is also a critical part of the SEF. Threat modeling allows the development groups to identify potential risks or attacks against an application even before it is built and to make decisions about how to address these risks.

Once identified, threats are ranked in importance and addressed according to a risk profile. Some threats should be addressed in the internal design of the component, product or solution; however, some threats can be addressed by proper configuration and integration, or might require additional components or management processes in order to adequately control risks. In many, if not all cases, there can be residual risk in deploying and operating the components, products, and solutions.

While it is not in the scope of this document to specify or document threat modeling, the general flow is as follows:

1. Identify the assets.
2. Identify the potential threats.
3. Assign an impact for each threat.
4. Determine the probability of compromise.
5. Rank the risks.
6. Define mitigating counter-measures as needed.

While threat modeling is often documented as a point-in-time step of the design process, incremental value is obtained if it is treated as a continuous process within the development cycle. The development team might want to revisit the risk assessment and threat model for each new release of software, or when new risks and threats are discovered.

Security requirements

Just like functional requirements and performance requirements, security requirements are needed to help ensure security is built into the application from the start. Security requirements define what new security features are required and how existing features should be changed to include necessary security properties. The objective of security requirements is to help ensure that the application can defend itself from attack.

The SEF suggests nine categories for security requirements and provides examples for each category. These include:

1. Auditing and logging
2. Authentication and authorization
3. Session management
4. Input validation and output encoding
5. Exception management
6. Cryptography and integrity
7. Data at rest

8. Data in motion
9. Configuration management

Together, these formulate the end-to-end security architecture for the product and thus should be considered alongside one another—not in isolation. Also, each of the categories has many sub-topics within it. For example, under authentication and authorization there are aspects of *discretionary access controls* and *mandatory access controls* to consider. Security policies for the product are an outcome of the implementation decisions made during development across these nine categories.

Many security requirements are generic across many types of applications: embedded systems, thick client software, and Web-enabled applications. It is important that the security requirements meet the business requirements of the software.

Secure coding

Most application security vulnerabilities typically are caused by one of three problems:

1. The requirements and design failed to include proper security.
2. During implementation, vulnerabilities were inadvertently or purposefully introduced in the code.
3. During deployment, a configuration setting did not match the requirements of the product on the deployment environment (for example, un-encrypted communication allowed over the Internet).

Attention to secure coding can prevent vulnerabilities being added during implementation. Secure coding guidelines are usually provided in a separate document that is specific to the development team's environment and chosen source code languages. Detailed information about topics including data validation, output encoding, handling of sensitive information, avoiding invention of encryption/decryption algorithms, exception handling, and source language-specific development tips, should be available to developers.

Use of automated security analysis tools is recommended, as is the use of proven certified security components. This allows developers to perform analysis of known security issues while emphasizing the use of secure and proven code components.

Test and vulnerability assessment

Testing applications for security defects should be an integral and organic part of any software testing process. During security testing, organizations should test to help ensure that the security requirements have been implemented and the product is free of vulnerabilities.

The SEF refers to the MITRE Common Weakness Enumeration⁵ (CWE) list and the Common Vulnerability Enumeration⁶ (CVE) list for the specific vulnerabilities for which products should be tested. This approach helps ensure that the SEF does not get stale with old vulnerability information and allows product development teams to reference a current list of weaknesses and vulnerabilities. This, in turn, can help ensure a relevant security assessment is performed against the most current set of known vulnerabilities.

Creating a security test plan is a critical part of test and vulnerability assessment. This test plan includes the documentation and analysis of several characteristics of the application:

⁵ For more information visit <http://cwe.mitre.org>

⁶ For more information visit <http://cve.mitre.org>

entry points, output locations, deployment environment, product functions and business logic, and application users, roles, and permissions. The SEF recommends performing security analysis using automation tools prior to IPD decision checkpoints, using the most current test cases and knowledge about threats and vulnerabilities.

The tools described in the following paragraphs should be used to perform automated analysis of source code, object code binaries, dynamic analysis and runtime analysis.

Source code security analyzers

These tools can analyze application source code to locate vulnerabilities and poor coding practices. These tools can also trace user input through the application (code flow analysis, taint propagation), to uncover various injection-based attacks. IBM Rational Software Analyzer and IBM Rational AppScan Source Edition are examples of these tools

Bytecode security analyzers

These tools can analyze application byte code (relevant for certain languages only), for the same vulnerabilities mentioned previously. In some scenarios, source code is not available to the tester, and bytecode can be used for the analysis.

Binary security analyzers

Binary analysis is very similar to source code analysis. However, instead of evaluating the source code, this analysis examines the application binary. When applications are compiled, the source code is interpreted by the compiler and is dependent on the environmental components that support it. This dependency on environmental factors can lead to contextual risks for some software deployments.

Dynamic analysis tools

These tools perform analysis of the application as a *black box*, without knowing its internal operation and source code. Dynamic analysis tools automatically map the application, its entry points and exit points, and attempt to inject input, which will either break the application or subvert its logic. IBM Rational AppScan Developer Edition is this type of tool.

Runtime analysis tools

Runtime analysis, strictly speaking, is not a specific security analysis technique or tool. Runtime analysis is the software development practice targeted at understanding software behavior during run time—including system monitoring, memory profiling, performance profiling, thread debugging, and code coverage analysis. Run-time analysis is almost always deployed in conjunction with another type of automated security analysis. For example, you might run a dynamic analysis tool against a Web-based application and monitor both the system resources for disk read/write as well as the application source code for code coverage. Viewing the application from both the dynamic analysis perspective (*black box*) and runtime analysis perspective (*white box*) during this process can lead to a greater understanding of the potential security issues that might exist, for example, stress testing and denial-of-service (DoS) attacks.

Documentation

The SEF not only documents *why* security documentation is essential, it provides guidance on *how* security documentation of the product should be structured.

Development projects within an organization should follow an Information Development Plan that outlines the required documentation for the individuals involved in the various roles of installing, configuring, operating, and managing the product or solution.

The SEF extends that requirement to consider security-related roles associated with the component or product. The security role definitions should include *security architect*, *enterprise architect*, *system integrator*, *system auditor*, and *product assurance evaluator*.

Further, the SEF recommends that Information Development Plans include considerations for security in the Integration, Deployment, Operations, and Management section of the documentation so that security remains visible and relevant over the range of expected deployment life cycles and roles. Security-related guidance in product documentation should include information about security-related settings for the underlying environment within which the software or solution will run.

Incident response

Product teams should follow a defined process for handling security-related incident reports. This process is put in place to help ensure that after an incident is discovered and validated, any other product teams in the organization that might be affected by the vulnerability are informed of the situation so they can begin working on a fix if necessary.

With complex combinations of component reuse and solution construction, such processes are necessary to ensure that potentially affected products are identified quickly.

Secure Engineering Framework summary

The Secure Engineering Framework applies additional security-related elements to all development process stages, thus allowing teams to apply these techniques to the particular development processes that they are following. The Secure Engineering Framework also integrates with a common development process, which product teams should follow for product development.

Continuous security improvement

Security is a moving target. The landscape of risks, threats, and vulnerabilities is continuously changing.

In the previous section we described the rationale for recommending that product development teams employ a structured development process as well as a Secure Engineering Framework. In addition, teams should recognize that these might not be sufficient for improving security on a continuous basis for existing and new projects. Often, project teams are implementing incremental features and components to large, complex solutions that are already in place. To address the complexity of this environment, the third pillar of secure product development is *continuous security improvement* (CSI).

Product teams should develop components, products, services, and solutions that are as free of security vulnerabilities as possible, and should strive to continually improve the security characteristics of these offerings.

This requirement defines a set of key performance indicators (that is, measurements/metrics) related to security characteristics of offerings. These measure security characteristics as well as performance of the offering team in achieving their goals for security throughout the offering life cycle.

Key performance objectives (that is, goals for these measurements/metrics) are set by the offering team and are based on accepted security practices within the organization and the

information technology industry, along with goals that demonstrate that the security characteristics of the offering will improve from release to release.

The metrics and progress in achieving the goals are to be reviewed throughout the life cycle of an offering. These reviews include:

- ▶ Establishing security quality and risk acceptance goals early in the offering planning process
- ▶ Project decision checkpoints in the development process
- ▶ Service quality reviews during the post-availability life cycle
- ▶ Selective management reviews
- ▶ Offering performance reviews

Key performance indicators

A continuous improvement framework for security should establish a set of indicators that represent tangible movement from a starting point to a desired state.

These *key performance indicators*, or KPIs, should capture the results of actions that represent security-oriented goals and achievements throughout the solution life cycle.

As a general rule, the KPIs for continuous security improvement fall into two categories:

1. Actions taken and results achieved in the development process, to be reviewed at time of availability.
2. Actions taken and results achieved in the post-release support and maintenance process, including serviceability performance and number of security defects.

In some cases, the continuous security improvement KPIs might be intuitive, for example, the number of security defects for a given release of software, or the time to diagnose and resolve a reported security defect. In other cases, the KPIs might be more subjective, for example, the number of security test cases executed and passed, or the percentage of source code that has undergone visual review.

The tables in the following sections highlight some examples of quantitative and qualitative key performance indicators that could be included in a continuous security improvement program.

Development process KPIs

The key performance indicators associated with the development process, to be reviewed prior to offering availability, are categorized by *quality*, *resilience*, and *integrity*.

- ▶ The pre-release *security quality KPIs*, shown in Table 1, are structured to promote best practices for security in development and the use of the best security functionality available in the offering operating environment.

Table 1 *Security quality KPIs*

Security quality KPI	Metric	Improvement trend
Proof that secure engineering practices were followed in development project.	List and work reports	Increasing
Comparison of offering integrity indicators with other offerings in hardware and software operating environment.	Higher / consistent / lower	Increasing
Comparison of offering resilience indicators in comparison to other offerings in hardware and software operating environment.	Higher / consistent / lower	Increasing
Use of the security features of the hardware and underlying software.	Use of best practice / rationale	Increasing

- ▶ The pre-release *security integrity KPIs*, shown in Table 2, are structured to promote offering assurance and integrity.

Table 2 *Security integrity KPIs*

Security integrity KPI	Metric	Improvement trend
Amount of developed components included in code review.	0% to 100%	Increasing
Amount of external components included in code review.	0% to 100%	Increasing
Amount of offering tested for known vulnerabilities.	0% to 100%	Increasing
Amount of offering with signed code and distribution packages.	0% to 100%	Increasing
Documentation for security features and standards.	0% to 100%	Increasing
Documentation for completed assurance review / regression tests.	0% to 100%	Increasing

- ▶ The pre-release *security resilience KPIs*, shown in Table 3, are structured to promote offerings that can be configured for resilient operation as they are deployed.

Table 3 *Security resilience KPIs*

Security resilience KPI	Metric	Improvement trend
Completed design documentation for resilient operation.	0% to 100%	Increasing
Completed deployment documentation for resilient operation.	0% to 100%	Increasing
Completed resilience testing (ethical hacking / penetration testing).	0% to 100%	Increasing

Support process KPIs

The key performance indicators associated with post-release support and maintenance track the security quality of the offering in operational environments.

- ▶ The post-release *security quality KPIs*, shown in Table 4, are structured to measure and track the number, type, and severity of security-related defects.

Table 4 Post-release security quality KPIs

Post-release security quality KPI	Metric	Improvement trend
Time to resolution for post-availability security problems (including CVEs).	Number in hours / days by incident	Decreasing
Percentage of fixes and changes that have undergone and passed code assurance review / regression tests.	0% to 100%	Increasing
Percentage of fixes and changes that have undergone and passed vulnerability tests.	0% to 100%	Increasing
Frequency of post-availability CVE reviews and retests.	Time between reviews	Decreasing

- ▶ The post-release *security serviceability KPIs*, shown in Table 5, are structured to measure and track the time to identify and resolve security-related defects.

Table 5 Post-release security serviceability KPIs

Post-release security serviceability KPI	Metric	Improvement trend
Number of post-availability security problems reported.	Number / Severity	Decreasing per release found
Number of post-availability CVEs published.	Number / Severity	Decreasing per release found

Offering-specific key performance objectives

Development teams should use the key performance indicators as guidance to set, track, and report on the actions and results related to security of the products and offerings.

At the start of each product delivery cycle, teams should set goals for their development and post-release serviceability KPIs. These goals or objectives are called *key performance objectives* (KPOs). The actions and results of the team are evaluated against the KPOs at various stages and milestones of the release life cycle. New offerings are expected to set initial KPOs consistent with the best practices in their development area.

The *development-oriented* key performance objectives include:

- ▶ Adoption of secure engineering development practices
- ▶ Code review coverage
- ▶ Extent of vulnerability analysis
- ▶ Depth of testing
- ▶ Defect remediation
- ▶ Documentation of security-related information
- ▶ Adoption of ecosystem security features
- ▶ Assurance testing

The *serviceability-oriented* key performance objectives include:

- ▶ Frequency of post-availability review of published vulnerabilities and exposures
- ▶ Fix code review
- ▶ Fix vulnerability testing
- ▶ Security regression testing
- ▶ Time to remediate and resolve security-related defects

Continuous security improvement summary

The *continuous security improvement* process is intended to help ensure that the security characteristics of product offerings improve over time and that security characteristics of new product offerings reflect best practices.

An assessment of the actions and the achievements of development teams with respect to the security of their offerings should be tracked and evaluated during and after offering availability. The review of the serviceability measurements of an offering helps to validate that attention was given to security in the development process. By having to meet goals that increase in each release, teams are compelled to continually improve their attention to security.

Supply chain security

IBM maintains one of the world's most recognized global supply chain management systems. IBM has received numerous awards for innovative supply chain management practices. For example, in 2008 IBM was recognized by the Supply-Chain Council for excellence in Supply Chain Research. In 2007 IBM was recognized by AMR as the supplier of the year.

IBM supply chain practices focus on effective management of product design, manufacturing, transportation, fulfillment, import and export, intellectual property management, and customer support. IBM has led the global focus on supply chain security and is a founding member of the Electronic Industry Supplier Code of Conduct. The IBM supply chain processes and policies are fully integrated with the standard product development and manufacturing process.

Supplier assurance

Before IBM conducts business with any external supplier, IBM Global Procurement has the responsibility to evaluate and assess the supplier to verify that they meet procurement criteria for qualified suppliers. These criteria include financial solvency, compliance with IBM technology and technical standards, and the ability to meet IBM's requirements.

This includes the following assessment:

- ▶ Ensure the supplier is not on an Unapproved or Denied Parties List.
- ▶ Supplier must commit to the IBM Supplier Conduct Principles.
- ▶ Suppliers providing hazardous waste, special waste, and end-of-life product disposal services must be in compliance with IBM Corporate instructions.
- ▶ Supplier must sign the IBM Security Letter Agreement (SECLA).

- ▶ Suppliers are required to submit to periodic assessments by responding to the Supplier Assessment Questionnaire.
- ▶ A supplier must submit to remediation actions if found to be out of compliance before being reinstated as an approved supplier.

An important element of the supplier assessment process is the *Supplier Risk Assessment*. The intent of the Supplier Risk Assessment is to identify all components that make up the overall supplier risk—offering, process, and business risks. Risk characteristics are identified to help assess the risk severity level. Mitigation strategies are also addressed as part of the assessment process.

Supplier conduct principles

Suppliers are required to adhere to the *IBM Supplier Conduct Principles*. More information can be found at <http://www.ibm.com/ibm/responsibility/supplychain.shtml>. Suppliers are assigned an overall rating score against their ability to deliver quality components or services in conjunction with continued compliance to supplier conduct principles. Suppliers are audited for compliance periodically. Suppliers found non-compliant are required to document an action plan to remediate. Suppliers found to be non-compliant are downgraded and might be placed on the unapproved supplier list.

Supply chain security policy enforcement

Suppliers who are directly or indirectly involved with tangible goods shipments to or on behalf of IBM are required to adhere to the *IBM Supply Chain Security* requirements. The IBM Security Letter Agreement (SECLA) is used to demonstrate commitment to the IBM Supply Chain Security Principles by suppliers that seek a relationship with IBM. This includes:

- ▶ OEM products that carry an IBM logo or are sold by IBM to be used in an IBM system (such as OEM feature cards, adapters, and so on) must also meet the same security, export, blue book policies and/or compliances that an internally built or “build to print” IBM system must meet. System integration and other testing in the development cycle to ensure proper function.
- ▶ Electronic components on an OEM subsystem are covered for quality and performance requirements through the statement of work (SOW), contracts, and other OEM specifications.
- ▶ It is the responsibility of an OEM provider to ensure the robustness, stability, performance, and ultimately the execution-time security of the software or firmware they deliver to IBM.
- ▶ Access to software or firmware development libraries (including firmware source code as well as documentation) is controlled by access control lists. Suppliers must be authorized by an IBM manager and must have a *need to know* before gaining access.
- ▶ Any firmware or software written for use by IBM is required to have a *Certificate of Originality* on file for every piece of open source or non-open source code that is picked up and incorporated into one of our deliverables. This is part of the release process for all firmware and software that is developed by IBM. Certificates of Originality are approved by legal and evaluated against stringent IBM criteria.
- ▶ All IBM developers are required to participate in open source training in order to produce code that is compliant with open source development guidelines. All firmware and software that is produced (internally and by suppliers) is run through an automated tool that identifies potential violations of this policy and each entry is reviewed by project managers and lawyers. This ensures that code contributions or adoption have been made

by *uncontaminated* developers and that code is free from potential intellectual property rights violations or inclusion of malicious components.

Employee and contractor assurance

IBM conducts a thorough background check of all suppliers and contractors and requires the same on behalf of approved suppliers. Before placement of supplier personnel at a customer site under a work authorization, for every person (including persons who are not U.S. citizens, green card holders, or permanent US residents), to the fullest extent permitted by applicable law, the supplier is required to perform or have performed a criminal background check covering the counties in which the person was employed or resided for the past seven years (or longer as required by applicable law). Suppliers are not permitted to propose persons who have had a serious criminal conviction (felony) or have been found guilty of an offense involving violence or dishonesty. A supplier relationship cannot be established with IBM unless the proper background checks have been completed within the past 3 years, (unless prohibited by applicable law), and are on file with the supplier.

The IBM supply chain management system is a model for organizations to follow. More details on the model can be found at:

- ▶ Main Supply Chain site:

<http://www.ibm.com/procurement/proweb.nsf/ContentDocsBytitle/United+States~Supply+chain+social+responsibility>

- ▶ Security in supply chain statement:

<http://www.ibm.com/procurement/proweb.nsf/ContentDocsBytitle/United+States~Supply+chain+security+requirement?OpenDocument&Parent=Information+for+suppliers>

- ▶ IBM Supply Chain Security Guidelines:

[http://www.ibm.com/procurement/proweb.nsf/objectdocswebview/filesupply+chain+security+guidelines/\\$file/supply+chain+security+guidelines+12sep03.pdf](http://www.ibm.com/procurement/proweb.nsf/objectdocswebview/filesupply+chain+security+guidelines/$file/supply+chain+security+guidelines+12sep03.pdf)

Summary

IBM has long been recognized as a producer of hardware, software, and solutions that are built with high quality, reliability, function, and integrity. IBM accomplishes this through attention to these aspects as teams conceive, design, develop, test, deliver, and service these offerings.

The Integrated Product Development (IPD) process, augmented with the Secure Engineering Framework (SEF), can help organizations ensure that appropriate attention to security is paid at all stages of product development. By following a direction of continuous security improvement, product teams work to continually improve the security characteristics of the hardware, software, and solutions we deliver.

The development process and offering life cycle within an organization are often elements of the larger global supply chain management system that ensures quality and integrity in the products and services that an organization provides worldwide. Attention to security is required across both the global supply chain and the development processes to deliver products that have appropriate security characteristics and resistance to vulnerabilities.

The team who wrote this guide

This guide was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO).

Danny Allan is director of security research with IBM Rational. Danny came to Rational through the acquisition of Web application security and compliance leader Watchfire in July 2007. He brings with him more than nine years of business and security technology-related experience, including penetration testing and internal system remediation for one of Canada's biggest universities. In his role as a security researcher, he is closely involved with enterprise global customer deployments, researching and evaluating technologies, and helping to define and recommend strategic directions. Danny has held several critical customer-facing positions, including Team Lead, Consulting Services, and Sales Engineer. He has published several white papers and articles and participates in industry working groups. He has also spoken at security events and is often called upon by key media including the Associated Press, Bloomberg, and the Wall Street Journal for his opinions regarding Web application security. Danny holds a Bachelor of Commerce degree with a major in Information Systems from Carleton University.

Tim Hahn is a Distinguished Engineer at IBM and has been with IBM for 19 years. He is the Chief Architect for Enterprise Modernization Tools within the IBM Software Group Rational organization. He is responsible for strategy and execution for the Rational Enterprise Modernization products that bring innovative and vibrant technology to meet the needs of a diverse user community focused on enterprise modernization. Tim previously worked in the IBM Software Group Tivoli® organization as the Chief Architect for Secure Systems and Networks, working on security product strategy, architecture, design, and development. Tim has worked on a variety of products in the past, including lead architecture, design, and development for the IBM Encryption Key Manager and the z/OS Security Server LDAP Server. Tim has published numerous articles discussing the usage of Rational and Tivoli security products in end-to-end deployment environments and is a co-author of two books: *e-Directories: Enterprise Software, Solutions, and Services*, Addison-Wesley, 2000, ISBN: 0201700395 and *Mainframe Basics for Security Professionals*, IBM Press, 2008, ISBN: 0131738569.

Andras Szakal is an IBM Distinguished Engineer and the Chief Architect of the IBM Federal Software business unit. Andras is an Open Group Distinguished Certified IT Architect, IBM Certified SOA Solution Designer, and a Certified Secure Software Lifecycle Professional (CSSLP). His responsibilities include developing e-Government software architectures using IBM middleware and leading the IBM U.S. Federal Software IT Architect Team. He holds undergraduate degrees in Biology and Computer Science and a Masters Degree in Computer Science from James Madison University. Andras has been a driving force behind the IBM adoption of federal government IT standards and is a member of the IBM Software Group Government Standards Strategy Team. His team has been responsible for helping the federal government move e-Government into the Smarter Planet™ era through the application of SOA and Cloud Computing. He is a member of the IBM Corporate Security Architecture Board focused on secure development and cybersecurity. Andras represents the IBM Software Group on the Board of Directors of The Open Group and currently holds the Chair of the IT Architect Profession Certification Standard (ITAC) within the Open Group.

Jim Whitmore is a Software Engineer at IBM and an Open Group Certified Lead Architect. He is currently responsible for advanced technology projects in the areas of Information Protection and Secure Cloud Computing. During his 25 years at IBM, Jim has led both networking and security-focused design and integration projects for clients in the government and industry sectors. In 2007 he was awarded a patent for a Security Design Methodology. Jim has been published in IBM Redbooks® and the IBM Systems Journal. He holds a BS in

Electrical Engineering and an MS in Telecommunications Management. Jim is a senior member of both the IEEE and the ACM.

Axel Buecker is a Certified Consulting Software IT Specialist at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on areas of Software Security Architecture and Network Computing Technologies. He holds a degree in Computer Science from the University of Bremen, Germany. He has 23 years of experience in a variety of areas related to Workstation and Systems Management, Network Computing, and e-business Solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Thanks to the following people for their contributions to this project:

Alison Chandler, Editor
ITSO, Poughkeepsie Center

Peter Bahrs, Simon Bodger, Gary Book, Bill Carpenter, Anne Dames, Walter Farrell, Matthew Flaherty, Pete Heyrman, Maryann Hondo, Takao Inouye, Julie King, Shawn Mullen, Chuck Murray, Linh Nguyen, Bill Odonnell, Harriet Pearson, William Penny, Ory Segal, Jeremy Shapiro, Adi Sharabani, Smriti Talwar, Nikola Vouk, Michael Waidner, Michael Weider, Douglas Weir
IBM

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts>
- ▶ Follow us on twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document, REDP-4641-00, was created or updated on March 18, 2010.




Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>



The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AppScan®	Redguide™	Tivoli®
IBM®	Redbooks (logo)  ®	z/OS®
Rational®	Smarter Planet™	
Redbooks®	System z®	

Other company, product, or service names may be trademarks or service marks of others.