# Security and Project Management

*Robert J. Ellison*

February 2006

ABSTRACT: Software errors can be introduced by disconnects and miscommunications during the planning, development, testing, and maintenance of the components. The likelihood of disconnects and miscommunications increases as more system components have to satisfy security requirements. Project managers should consider the additional communications requirements, linkage among life-cycle activities, and the potential usage environment as these items relate to security needs.

## BUSINESS CASE

An organization can either incorporate security guidance into its general project management processes or react to security failures. It is increasingly difficult to respond to new threats by simply adding new security controls. Security control is no longer centralized at the perimeter. Meeting security requirements now depends on the coordinated actions of multiple security devices, applications and supporting infrastructure, end users, and system operations. Reengineering a system to incorporate security is a time consuming and expensive alternative.

A recent Computer World article quoted Theresa Lanowitz, an analyst at Gartner Inc. [Hildreth 05]:

The life cycle may appear obvious, but most organizations—close to about 90%—do not know how to effectively manage the life cycle. If the life cycle was truly embraced with the right people, process, and technologies, we would see better-quality software and more efficient and effective IT organizations. As it is, most IT organizations waste quite a bit of their budget because they have bad business practices, fail to deliver on requirements, and fail to manage projects to meet schedule, cost, and quality goals.

On the list of examples of software failures for the Computer World article was "A software bug apparently caused the largest power outage in North America, the Northeast blackout of August 2003, which threw millions of people into darkness." The analysis of that event, though, identified a collection of system, organizational, and operational errors [NERC 04]. The software error was certainly one trigger for the incident, but the eventual failure of the power grid was the result of multiple of errors in system development and in operations.

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

Cyber attacks take advantage of software errors, such as not properly validating user input, inconsistencies in the design assumptions among system components, and unanticipated user and operator actions. Software errors can be introduced by disconnects and miscommunications during the planning, development, testing, and maintenance of the components. Although an application development team may be expert in the required business functionality, that team usually has limited or no applicable security expertise.

The likelihood of disconnects and miscommunications increases as more system components have to satisfy security requirements. The necessary communications and linkages among the life-cycle activities, among multiple development teams, and between the system development and eventual usage should be reflected in project management. Project managers should consider the additional communications requirements, linkage among life-cycle activities, and the potential usage environment as these items relate to security needs.

## OVERVIEW

The Trustworthy Computing Security Development Lifecycle provides an example of a pragmatic way to incorporate security into development [Lipner 05]. The objective of the SDL is not to overhaul an existing process totally but to add well-defined security checkpoints and security deliverables.

This note shares the Microsoft objective to enhance an existing process by describing the security role for project checkpoints and deliverables, as well as discussing how security requirements affect project planning and monitoring. The assumption is that the reader is an experienced manager but has limited security knowledge. For organizations moving to make security a higher priority, project managers need to address how that change affects the following:

- requirements and scope
- the technical plan
  - project life cycle (deliverables and sequencing of deliverables)
  - activities required to complete deliverables
- resources
  - skills needed
  - facilities, tools
- estimates
  - duration of resource requirements
  - other related estimates such as size and defects
- project and product risks

## PROJECT REQUIREMENTS AND SCOPE

Security's impact on scope has several dimensions. The scope is influenced by the type and number of threats, by the sophistication and resources available to the attacker, by the desired response to an attack, and by the level of assurance required that the system meets its security requirements.

A risk assessment should aid in identifying the highest priority threats and the profiles of the most likely attackers. Straightforward preventive measures may offer sufficient protection from the inexperienced attacker. Experienced and well-resourced external attackers and "insiders" require more elaborate tactics.

The scope is influenced by the desired response to attack. A passive response does not depend on the system having knowledge of an attack and is typically preventive. For example, input validation is a passive response that prevents a significant number of attacks. An active response is an action that takes place when a fault is detected. An active response that improves reliability in the event of a hardware failure would be automatic failover of processing to a redundant system. A simple active response for security might be an automatic system shutdown when an attack is detected to protect resources, but a more frequent objective for an active response is to continue to provide essential services during an attack by dynamically changing system behavior. Hence, an active response typically increases software complexity.

The level of assurance required affects all aspects of project management. We do not attempt to define those levels. In practice, the assurance level depends on the consequences of a security failure. The issues associated with high assurance systems have received attention because of their importance for national defense and for domains such as medicine and nuclear power. The project summaries collected by Yen and Paul identify commonalities among high assurance applications in diverse domains [Yen 98]. Fred Cohen, in a Burton Group presentation at Catalyst 2005, described medium risks as those for which the consequences could reasonably lead to substantial reduction in shareholder value, confidential business information to be leaked, legal liability above normal business liability insurance, or substantial civil action or negative publicity. Medium assurance could be applicable to corporate financial systems, manufacturing control systems, and the information systems used for critical infrastructure services such as power and water.

Access to corporate information may have to satisfy legal, regulatory, or fiduciary duties, contractual obligations, or voluntary requirements such as the protection of proprietary data. Those requirements raise the importance of security governance, i.e., the incorporation of security into business management. Security governance is typically associated with systems that require medium or higher

assurance. Successful security governance depends on developing control and feedback structures.

Regulatory compliance may depend on formalizing governance and risk management and, for each regulation, may require specifying the scope in terms of the responsibilities and roles for personnel and IT systems.

## TECHNICAL PLAN

The nature of threats and their consequences affects both planning and resources. The mitigation of low consequence and low likelihood threats might be left to the discretion of the project leader with limited management review. The management of high probability threats with medium level consequences would likely require external expert assistance and a well-defined systematic review process.

Testing is also influenced by the risks. See the Security Testing content area for a discussion of security test planning.

### Project Life Cycle (Deliverables and Sequencing of Deliverables)

Risk analysis should be a thread through the development process and hence provides an indirect measure of how well potential errors have been analyzed and then addressed. There should be a close tie between the outcome of risk analysis and requirements as risk analysis helps to define the scope for security in terms of the threats to be considered, the response desired, and the assurance level required for that response.

### Risk and Threat Analysis

Threats are the potential attackers and are described in terms of an actor (employee, business partner, contractor, outsider) with an objective (financial gain, obtaining proprietary corporate information, disabling essential business systems), and with a set of resources (funding, personnel, computing hardware, skill, knowledge of internal systems).

A risk assessment explores how a component could be exploited by the identified threats (i.e., what could go wrong) and analyzes the possible responses to such attacks. The response options for a risk are to (a) mitigate (reduce probability of event, reduce impact, improve recovery), (b) transfer (insurance, contracted agreements), (c) ignore (for low impact and highly unlikely threats), or (d) avoid, which may require changes in requirements. The factors involved with a risk assessment that is done early in the development process are predominantly

business rather than technical. Project management needs to ensure stakeholder participation in such activities. (See the Architectural Risk Analysis content area.) The attack patterns would be rather abstract for a preliminary system risk assessment and would become more detailed as the software architecture and detailed design are created.

Architectural risk analysis is an example of an important security checkpoint. The software architecture describes the system structure in terms of components and specified interactions. The increased system specificity provided by the software architecture also increases the specificity of the threats and the desired system response. An architectural risk assessment can review the threats, analyze how the architecture responds to those threats, and identify additional threats introduced by that architecture. (See Risk Management Framework and Architectural Risk Analysis. Also see the Assembly, Integration & Evolution content area for a more detailed discussion of integration issues.)

Table 1 lists a number of software assurance checkpoints that should be incorporated into the project plan. The implementation of these checkpoints depends on the characteristics of the software. The risk analysis for an integrated system has different requirements from the risk assessment of a commercial product or an infrastructure component. The differences in software assurance issues and project management guidance among products, application/integrated systems, and systems of systems are discussed in The Influence of System Properties on Software Assurance and Project Management.

*Table 1. Examples of Security Activities and Checkpoints*

| | |
|---|---|
| Sysstem Risk Analysis | Defines the scope for security<br>Provides<br>• relative importance of assets and business activities (integrity, availability, confidentiality)<br>• relative importance of attacker characteristics (insider, customer, business supplier, skilled, general Internet virus or worm)<br>• desired response: mitigate, transfer, ignore, avoid |
| Architectural Risk | Assets<br>• Risk and Response Model: Describes likely attacks based on the design proposed by the architecture. Attacker profiles, attack targets, and proposed system response should be consistent with those established by the initial risk analysis or requirement elicitation. Essential component for an assurance case.<br>• Architecture execution view: Runtime decomposition of system into components.<br>Security issues include the following:<br>• Have the highest risk attacks been identified?<br>• Is the proposed response appropriate, and does the architecture implement that response in an effective way? |

| | |
|---|---|
| | • Are there features of the architecture that raise the security risk for the deployed system or that have security risks that would be difficult or expensive to mediate? |
| Component Development | Assets:<br>• Architecture<br>A large or complex component may require architectural risk assessment.<br>• Risk and Response Model<br>Describe risks and responses with respect to component architecture and design. Maintain consistency with system risk and response model. Provide detailed support for an assurance case.<br>• Design<br>Design review: Demonstrate design consistent with identified risks.<br>• Source code<br>Use static analysis tools to demonstrate the absence of classes of coding errors.<br>• Executable component<br>Fuzz testing: Microsoft claims the technique is effective in identifying errors that could lead to vulnerabilities [Lipner 05]. The appearance of a significant number of errors during fuzz testing is an indicator of generally poor software quality. (See the Black Box Testing content for a discussion of fuzz testing and related techniques.)<br>• Risk-based testing: Risk assessment and threat modeling should identify protocols, components such as COTS software, and specific functionality that are security risks. Identify ways to stress those items. (See White Box Testing and Security Testing content.) |

### Activities Required to Complete Deliverables

Regulatory or contractual compliance may require demonstration that the software provides the necessary controls for accessing the information (i.e., the production of an assurance case). Security governance typically increases the complexity for meeting security requirements. For example, business process compliance may require showing that the composition and interactions of multiple applications maintain the required controls and feedback.

Delivering "secure" software requires demonstrating that the desired level of assurance has been achieved. While demonstrating that a system provides the required functionality is an essential aspect of software assurance, software security assurance depends more on demonstrating what a system does not do. Does improper input lead to a system failure or enable an attacker to bypass authentication or authorization defenses? (See the Black Box Testing, Security Testing, and White Box Testing content areas.)

The production of such an assurance case must be planned and managed. An assurance case provides an argument for how the software meets an identified threat. That argument typically is based on assumptions about how the software

behaves under certain operating conditions. Hence, an early step in building an assurance case is to provide evidence that software behavior satisfies the assumptions of the assurance argument. The production of an assurance case is an incremental activity. The assurance case should describe the architecture's role for meeting security requirements, and the architectural risk assessment and analysis should provide evidence that the architecture satisfies those requirements.

The activities listed in Table 1 can be part of an assurance argument. Syntactic analysis of the source code reduces the probability of coding errors that might lead to a security vulnerability. Risk-based testing can target the components and interfaces that are most likely to lead to a system compromise.

An assurance case may be part of the requirements for contracted development. How will the assurance of delivered software be demonstrated? Do the assurance cases for the supplied software support the assurance argument for the integrated systems?

## RESOURCES

### Tools

The development environment requires a level of security commensurate with the planned security level of the product being produced. Appropriate controls and configuration management of the development artifacts are essential. There may be specific tools required, such as for static code analysis, to aid the production or testing of secure software.

As assurance levels rise, the development process should provide the necessary control and information protection mechanisms. Change management must be well controlled. High-assurance configuration management must support requirements for audit, traceability, and process enforcement. For very sensitive code segments, security governance may require that changes always be made by two developers to limit the ability of an individual to insert malicious code.

### Facilities and Staffing

Security expertise on most projects is limited and may be an internal or a contracted service. The allocation of that resource is often difficult even when security activity is limited to networks, authentication, and access control, but when security has to be incorporated into application development, that expertise is

spread much thinner. An increase in the level of assurance can significantly affect the both the security and software engineering expertise required.

For this discussion, we divide security expertise into two categories. One category consists of knowledge of security functionality such as the specification and implementation of access control, authentication, and encryption functions. Such security functionality may be encapsulated in the system infrastructure. The second category of expertise consists of the skills to identify and mitigate exploitable system vulnerabilities. Historically, a significant number of the vulnerabilities that lead to a security failure were created by application errors and not by failures with the security infrastructure. Vulnerabilities may be in the least exercised parts of the system and depend on pathological aspects of the interface. Such vulnerabilities may be missed by application development teams, who normally concentrate on the core functionality.

The security functionality for authentication, authorization, and encryption is typically composed of commercially supplied components that can be tailored for a specific operating environment. Those components must have the required assurance level. It would not be surprising to find the security knowledge associated with the first category to be concentrated within a few teams. The security specialists associated with that infrastructure should be aware of the security issues associated with development and project management. Unfortunately, application development teams rarely have the necessary security expertise. The resources in the second security knowledge category must be spread across multiple development efforts.

Microsoft's experience with the implementation of The Trustworthy Computing Security Development Lifecycle is that someone with security expertise must be available for frequent interactions during software design and development. A similar recommendation has been given for agile development [Wäyrynen 04]. Microsoft created a central security group that drives the development and evolution of security best practices and process improvements, serves as a source of expertise for the organization as a whole, and performs a final security review before software is released. For example, during the requirements phase, the product team requests the assignment of a security advisor from the central group who serves as point of contact, resource, and guide as planning proceeds. The security advisor helps the product team by reviewing plans, making recommendations, and ensuring that the central security team plans appropriate resources to support the product team's schedule. The security advisor makes recommendations to the product team on the security milestones and exit criteria that will be required based on project size, complexity, and risk.

Tasks such as risk assessments, code reviews, and threat modeling require security expertise. On the other hand, there are security improvement practices that can be implemented without requiring extensive security experience. For example, although security knowledge may be necessary to configure a tool for the static analysis of the source code, the use of such a tool does not require a security background. (See the Code Analysis Tools content area.) Testing provides a second example. Penetration testing is often part of an acceptance test or certification process. Penetration testing might be implemented by what is called a red team: security experts who attempt to breach the system defenses. Fuzz testing is a simple form of penetration testing that finds software defects by feeding purposely invalid and ill-formed data as input to program interfaces [Arkin 05, Lipner 05]. Fuzz testing does not replace the need for testing that targets explicit security risks, but it is an example of an approach that can be used without detailed knowledge of security vulnerabilities. (See the Black Box Testing Tools content area for a discussion of the effective use of fuzz testing.)

## ESTIMATES

An increase in the required assurance level can have a significant impact on costs and schedules, as such a change affects the development skills required, the tool support, development practices, and the procedures required to demonstrate that assurance. (See Business Case content.) Cost-saving strategies such as reuse of existing components or general-purpose commercial components may not be applicable for medium- and high-assurance systems.

The early estimates for effort, damage, and preventive costs have large variances. A vulnerability analysis model with more detailed attacker actions and possible responses requires a more detailed description of the software such as that provided by the software architecture or a detailed design.

Shared infrastructure can reduce component development costs, but those shared services typically aggregate risks. Estimates should reflect the increased assurance that can be applied to the shared services.

### Duration of Resource Requirements

Security is a concern throughout development. Risk analysis and mitigation have to be closely coupled with business risks and business operations. Hence, that connection must be maintained over the duration of the project.

The nature of the security expertise required obviously varies over the development life cycle. General security expertise might be stretched thin in the initial

planning and requirements phases when teams without that experience will require assistance. The planning for security testing should start after the architecture is defined. Risk analysis has to be a continuing activity but the specific expertise required may vary. Architectural risk analysis can take advantage of both domain and a breadth of architectural experience. The analysis of a detailed design may require in-depth knowledge of a specific technology, while the analysis of an implementation draws on a detailed knowledge of known exploits.

### Other Related Estimates Such as Size and Defects

Software vulnerabilities may be intentionally inserted during in-house or contracted development. These vulnerabilities can be much more difficult to find. Change and configuration management procedures provide some assurance for internal development.

Some security risks are inherent in the operating environment or with the desired functionality and hence are unavoidable. For example, it may be very difficult to block a well-resourced denial-of-service attack. Other risks may arise because of the tradeoffs made. A corporation may decide to allow employee access to corporate assets with computing equipment such as laptops or PDAs that are not managed by the organization.

The types of defects depend in part on the development context. Security failures have frequently been traced to coding errors such as a buffer overflow. From the perspective of such coding errors, improved code reviews and the use of static analysis tools should reduce those kinds of component errors. (See the Code Analysis Best Practices, Coding Practices, Coding Rules, Guidelines, and Code Analysis Tools content areas.)

## PROJECT AND PRODUCT RISKS

### Scope

Poor management of requirements scope is another frequent cause for project failure. Scope management is particularly important where the learning curve is a necessity because of the immaturity of the business usage or the supporting technology. Business integration requirements are pushing the connectivity of networked information systems beyond an organization’s IT systems. Meeting business requirements may depend on using relatively new protocols such as those for Web Services. Those protocols are currently a moving target, as they continue to be revised to reflect the experiences of early adopters. Best practices in this context have short lives, and the lack of well-defined and proven practices

adversely affects planning. Plans for these circumstances might include a prototype or use of an iterative or incremental approach.

Scope, as discussed earlier in this note, has multiple dimensions. Unfortunately, requirements may omit some of those dimensions. Potential requirements for secure data access during development, secure facilities, or demonstration of capability can add great complexity and schedule concerns to projects.

### Added Risks

Security mechanisms that mitigate a specific risk may create additional ones. For example, security requirements for managing identity for a large distributed system might be met by implementing authentication and authorization as infrastructure services shared by all applications, but the aggregation of authentication and authorization mechanisms into a shared service makes that service a single point of failure and a possible attack target. Such design decisions should involve a risk assessment to identify any new threats that require mediation, as well as the analysis of the operational costs after the system is deployed.

## SUMMARY

Clearly system security affects many of the "knowledge areas" of project management: specifically, scoping, human resources, communications, risk management, procurement, quality, and integration.

Providing the necessary level of security assurance requires more than the development of what is usually called the security architecture: perimeter defenses (firewalls), proxies, authentication, and access controls. An objective for the Chief Information Security Officer of one Wall Street investment house is to empty that security architecture (i.e., avoid treating security as an add-on) and instead to "raise the bar" for component software assurance by integrating assurance into the development processes. Such integration has to be reflected in project management.

Activities such as an architectural risk assessment, threat analysis, and static analysis for the source code provide checkpoints for specific development phases. Development controls and change management are essential development tools. However, the software assurance issues during development are dynamic, and project management must maintain linkages between business and technical perspectives, among life-cycle phases, and among development teams. The production of an assurance case can serve as an integrating mechanism by identify-

ing threats and desired responses and then tracing and refining the threats and responses during development.

A change in the level of assurance required can significantly affect the management of a project. Does the development staff have the requisite skills? How can that assurance be demonstrated? Can the existing software practices provide that level of assurance? This site provides a starting point for a discussion of best practices with respect to software assurance.

## BIBLIOGRAPHY


[Arkin 05] Arkin, B.; Stender, S.; & McGraw, G. "Software Penetration Testing." Security & Privacy Magazine 3, 1 (Jan-Feb 2005): 84–87.

[Berkun 05] Berkun, Scott. The Art of Project Management. North Sebastopol, CA: O'Reilly, 2005.

[Beznosov 04] Beznosov, Konstantin. "Extreme Security Engineering: On Employing XP Practices to Achieve 'Good Enough Security' without Defining It." First ACM Workshop on Business Driven Security Engineering (BizSec). Fairfax, VA, Oct. 31, 2003.

[Beznosov 05] Beznosov, Konstantin & Kruchten, Phillipe. "Towards Agile Security Assurance," 47–54. Proceedings of the 2004 Workshop on New Security Paradigms. White Point Beach Resort, Nova Scotia, Canada, September 20-23, 2004. New York, NY: Association for Computing Machinery, 2005.

[Boehm 03a] Boehm, Barry & Turner, Richard. Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Addison Wesley, 2003.

[Boehm 03b] Boehm, B. & Turner, R. "Using Risk to Balance Agile and Plan-Driven Methods." Computer 36, 6 (June 2003): 57-66.

[Brooks 95] Brooks, Frederick P. The Mythical Man-Month: Essays on Software Engineering. Reading, MA: Addison Wesley, 1995.

[Gall 02] Gall, John. The Systems Bible. Walker, MN: The General Systemmantics Press, 2002.

[Hildreth 05] Hildreth, Sue. "Buggy Software: Up From a Low-Quality Quagmire." Computerworld, July 25, 2005.

[Leveson 05] Leveson, Nancy. "A Systems-Theoretic Approach to Safety in Software-Intensive Systems." IEEE Transactions on Dependable and Secure Computing 1, 1 (January-March 2004): 66-86.

[Lipner 05] Lipner, Steve & Howard, Michael. The Trustworthy Computing Security Development Lifecycle (March 2005).

[Maier 96] Maier, Mark W. Architecting Principles for Systems-of-Systems (1996).

[NERC 04] North American Electric Reliability Council. August 14, 2003, Blackout: Final NERC Report (2004).

[Neumann 04] Neumann, Peter G. Principled Assuredly Trustworthy Composable Architectures (Final Report to DARPA, CDRL A001). Menlo Park, CA: Computer Science Laboratory, SRI International, December, 28, 2004.

[Wäyrynen 04] Wäyrynen, J.; Bodén, M.; & Boström, G. "Security Engineering and eXtreme Programming: An Impossible Marriage?" 117-128. Extreme Programming and Agile Methods-XP/Agile Universe 2004 (LNCS 3134). Edited by C. Zannier, H. Erdogmus, and L. Lindstrom. Berlin: Springer-Verlag, 2004.

[Weinberg 75] Weinberg, Gerald. An Introduction to General Systems Thinking. New York, NY: John Wiley & Sons, 1975.

[Weinberg 92] Weinberg, Gerald. Quality Software Management: Volume 1, Systems Thinking. New York, NY: Dorset House Publishing, 1992.

[Yen 98] Yen, I-Ling & Paul, Ray. "Key Applications for High-Assurance Systems." Computer 31, 4 (April 1998): 35-46.

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



DM-0001120