Chapter XIX
Security Considerations in the Development Life Cycle

Kenneth J. Knapp
U.S.A.F. Academy, USA

ABSTRACT

To promote the development of inherently secure software, this chapter describes various strategies and techniques for integrating security requirements into the systems development life cycle (SDLC). For each major phase of the SDLC, recommendations are made to promote the development of secure information systems. In brief, developers should identify risks, document initial requirements early, and stress the importance of security during each phase of the SDLC. Security concerns are then offered for less traditional models of systems analysis and development. Before concluding, future trends are discussed. Practitioners who read this chapter will be better equipped to improve their methodological processes by addressing security requirements in their development efforts.

INTRODUCTION

A perception exists among some information system (IS) security professionals that systems developers generally do not consider security as an integral part of the development process. Instead, a perception exists that developers often treat security more as an afterthought. Considering today's high-threat cyber environment, it is essential that security requirements remain a priority throughout the systems development life cycle (SDLC). In this paper, a description of SDLC strategies and techniques is provided to promote the development of secure systems. The key to integrating security into the SDLC is by documenting security requirements early and making security considerations a priority during each phase of development. Practitioners who read this chapter will be equipped to improve their methodological processes by including security requirements in their development efforts.

BACKGROUND

The motivation to write this chapter initiated from a previous study that involved a set of interviews the author conducted with certified information security professions between 2004 and 2006. In 2004, the

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

author conducted e-mail interviews with over 200 certified information systems security professionals (CISSP) located in over 20 countries worldwide. The interviews discussed what each participant felt were the most critical information security issues facing organizations today. A recurring perception among the participants was that security concerns are not a high priority among many system developers. One typical response stated, "It seems that unless the project is a security initiative, the involvement of a security resource to identify control requirements is an afterthought. By the time the security resource is formally involved, the project is so far ahead that insisting on changes to accommodate [security] controls is often viewed as a source of threat to the project's timelines." Another expressed frustration by stating, "Vendors are under constant pressure to meet their figures [deadlines], thus delivering the product with sacrifices in security and quality. The end customer assumes the risk when this software is delivered into the market space. This is a great concern these days and will continue to be a major security concern even 10 years from now." Another interviewee said, "Late security planning is (often) started well into the implementation phase of a project's SDLC. A current large organization is deploying a…management enterprise tool. This project has been in engineering and deployment for over a year and they are just now beginning to supply security."[1]

In 2006, the author conducted an interview with a certified information security professional with over 20 years of IT experience working for both government and Fortune 100 employers. The interviewee indicated that his own company recently cut security engineers by 50% in an effort to save costs and that top management typically does not place much priority on security unless they can be convinced that security requirements effect the financial bottom-line. Yet, demonstrating how security can bring about direct financial benefits or improve return on investment can be challenging for security professionals. Many find it difficult to quantify security in financial terms and stumble over simple questions asked by top managers, such as, "What am I getting for my security dollars?" One study found that U.S. companies spent only an average of .047% of their revenue on security (Geer, Hoo, & Jaquity, 2003). The difficulty of quantifying security benefits in financial terms that top management can appreciate is partially to blame for these low budgets.

One of the reasons that security is perceived as a development afterthought is that modern software systems often contain serious security flaws and require frequent patch updates. Yet, evidence suggests that by focusing on building security into software during development, money can be saved by minimizing the number of flaws that require patch updates. A 1981 IBM Systems Sciences Institute study reported that the cost to fix an error found after product release can be 100 times more costly than one identified during the design phase (Geer, 2002). For security defects, late fixes often cost more because in addition to having to remediate the flaw, successful exploits may lead to data theft, sabotage, or other cyber-related attacks (Berg, 2006). Security professionals should keep these facts in mind when asked by top management, "What am I getting for my security dollars?"

To promote the development of inherently secure software, this chapter describes various strategies and techniques of integrating security requirements into the SDLC. Secure software does not happen by itself, it requires process improvement and commitment from the development team and from management. The processes discussed in this chapter do not require the creation of a separate development process; instead, the processes can be integrated into an organization's existing methodology.

SECURITY CONSIDERATIONS IN THE SYSTEM DEVELOPMENT LIFE CYCLE

This section describes important security considerations for the major phases in a representative software development life cycle model. This model emphasizes developing information systems in general stages that often overlap, but follow a progressive development toward a fielded system.[2] For the purposes of discussing security considerations during software development, the following paragraphs address strategies and techniques for incorporating security in a notional seven-phase model that is an expanded version of a model presented in Bishop (2005). The seven phases covered in this paper include: system investigation, requirements analysis, system and software design, implementation and unit testing, integration and system testing, operation and maintenance, and retirement. This section will also discuss the importance of regular reviews as well as considerations for less traditional models of system development.

System Investigation

System developers should not necessarily use all the security measures discussed in this chapter because not all systems require the same level of security. Just as an automobile owner would not install an expensive car alarm in a used automobile with little worth, system developers should not build high security into a system that poses little risk. Since a high-security system can entail additional development and administrative costs, the level of security should address the level of estimated risk. The more sensitive and valuable the data within a system, the higher the priority security should have during development. In short, the security level of the proposed system should be sensible and support the business goals of the organization. Supporting this goal, the initial security assessment serves as a guide to ensure the security level in the system is reasonable and not overly burdensome.

In this first phase, analysts investigate the feasibility of a proposed system to include an initial estimation of cost and time. To promote security, this phase can include an *initial systems risk assessment* that developers can use as a basis for future security decisions. Emerging from this assessment, the proposed system can be generally classified as a low, medium, or high-risk system. The risk level should drive the designed level of security in the proposed system. This assessment can include an analysis of the target environment, an identification of threat vulnerabilities, as well as the potential of undesirable outcomes. This assessment can also evaluate potential countermeasures (Hansche, Berti, & Hare, 2004). The initial security assessment is a necessary step before moving to the more detailed phases of the SDLC.

Requirements Analysis

During this phase, initial requirements documented in the investigation phase are expanded to describe the intended functionality of the proposed system. These functional requirements can include a number of important security considerations. Following is a list of eight categories that developers should address during this phase:

- System management processes such as security training and policy requirements
- Application processing and data storage security requirements
- Access control and system sign-on requirements
- Transmission, communication, cryptographic, and network security requirements
- Physical and facility security requirements
- Operational and administrative security requirements
- Disaster recovery planning and system continuity requirements
- Ethical, legal, and regulatory requirements

The requirements should be collected from the functional users, system owners, and qualified experts in their respective area (e.g., a lawyer can help define the legal requirements). At this point, it is important that the output from this phase does *not* provide specific technical or engineered solutions. The problem of jumping directly into technical solutions is that the developers may not understand the essential security requirements of the proposed system. Without this knowledge, it is more likely the developed system may not have the optimum degree of security; the system may have too much or, more likely, too little security. For example, while documenting data communication needs, developers can write the functional requirements that data needs to be protected using "state of the art technology" or "strong encryption" rather than recommending a specific cryptographic solution. Recommending specific product solutions should be a part of the design phase.

Systems Security Requirements Document

For some systems and particularly higher risk systems, it might be practical to weigh the option of producing a system security requirements document (SSRD). A stand-alone SSRD has the benefit of emphasizing the importance of documenting security requirements so that security does not get lost in the primary document. However, in some development cultures, a separate SSRD may actually increase the risk that developers treat security requirements separately and not part of the overall development strategy. This can be a problem if developing a separate security document means that security is "out of sight" when important design decisions are made (Flechais, Sasse, & Hailes, 2003). Nevertheless, depending on the risk level of the proposed system and the culture of the development environment, developers should weigh the merits of writing an SSRD.

Security Requirements Traceability

It is valuable to extend the concept of requirements traceability to track security requirements throughout the SDLC. Requirements traceability is intended to ensure continued alignment between user requirements and various outputs of the systems development process (Ramesh & Jarke, 2001). Each identified requirement should have a specific solution in the design of the system and be tested during the appropriate phase of the SDLC. Likewise, each design feature should be traceable to an identified requirement. Requirements traceability can help designers identify areas in a developing system that do not meet the documented requirements. Thus, applying requirements traceability can help developers integrate security requirements throughout the SDLC.

System and Software Design

Documentation produced during the system and software design phase includes the technical and engineering designs that are traceable to the original requirements. Here, developers specifically spell out technical solutions, such as proposing a network architectural design or a cryptographic product. It is critical that the proposed design adequately addresses security requirements. If requirements are neglected, the system may not provide adequate security resulting in costly configuration changes and redesign efforts once the system is fielded. Security should not be treated as a design afterthought; instead, each identified requirement should have a traceable solution that is built into the system design.

For higher risk systems, in addition to unit and system test plans, developers should consider writing a dedicated security test plan during this phase. A test plan involves defining the criteria by which the system will be tested and measuring the criteria against an acceptable failure rate (Ayer & Patrinostro, 1992). Such a document spells out exactly the type of tests that are necessary to demonstrate that individual program units, as well as the total system, meet the original security requirements. Such a test document is advantageous as it objectively lays out test requirements during the design phase before system implementation begins. A test document can also have benefits during certification & accreditation of the system, which is discussed shortly.

Implementation and Unit Testing

In this phase, developers implement and test the software programs built during the previous design phase. Experienced testers realize that even the best system designs and most competent developers can produce systems with security flaws. The purpose of testing is to identify and correct these flaws before system operation. During unit testing, system testers should ensure that security solutions meet the original requirements. Tests can include both manual and automated code reviews that look for common design flaws and program errors. Some of the more common security flaws include unnecessary design complexity, buffer overflows, or incomplete error handling (Murray, 2001). Some automated tools will assess a piece of code without requiring a complete and integrated application. Thus, coders and testers alike can use tools to pinpoint vulnerabilities at the precise line of code and apply a fix early in the SDLC (Berg, 2006). In higher-risk systems, consideration can be given to hiring an independent agency to objectively and thoroughly conduct unit testing. Finally, testers should take care to use fabricated and not sensitive data in testing environments. For example, using actual social security numbers should be avoided during testing.

It is important to note that during this phase, implementation and testing can be treated as separate phases or can be accomplished concurrently. Either way, cooperation and teamwork between coders and testers will best ensure the correction of identified flaws. To this end, it may be advantageous to use some type of defect reporting and tracking database. Such a tool allows the testing team to enter identified flaws for coders to correct (Jordan, 2006). The tool should identify each flaw and provide a history of the flaw until the corrected code is successfully re-tested. To be thorough, the flaw history should document all retests after each attempt to fix any particular flaw.

Integration and System Testing

During integration, the development team combines individual program units into a total system. Testers then ensure that the completed system meets its intended security requirements. Like unit testing, this process can be iterative as testers find bugs for the development team to correct. The corrected system is then re-tested. Testing can be done using automated tools as well as by manual procedures. Because some security flaws are difficult for humans to find, automated tools are available to assistant testers during the testing phases of the SDLC.

Abuse Cases

A practical approach to system testing involves a process called *abuse cases*, which is a concept derived from *use case* development. Use cases are a behaviorally related sequence of steps for the purpose of completing a single business task or case (Whitten, Bentley, & Dittman, 2000). A use case is not a functional requirement, but instead a story of how users will use the system in their day-to-day activities. An *abuse case* is similar except that developers document ways that the system can be maliciously used. Abuse cases explicitly test for security flaws. For example, the developer can create a series of abuse cases that depict how malicious users can seek to overflow input buffers, insert malicious logic, and penetrate system defenses. Once fielded, the system can be retested against these cases to see if the system is adequately protected (van Wyk & McGraw, 2005).

Certification and Accreditation (C&A)

Completed systems can go through a mandatory C&A assessment that includes a security evaluation of the system against a predetermined set of standards. Certification is the technical evaluation of compliance with security requirements for the purpose of accreditation. Security accreditation is the official management decision given by a senior agency to authorize operation and explicitly accept the risk of an IS based on agreed-upon controls. By accrediting an information system, management accepts responsibility for the system and is accountable for adverse impacts (Hansche et al., 2004). An important C&A milestone can help developers as well as management to focus and maintain a security-aware mindset throughout all phases of the SDLC.

Operation and Maintenance

Once integrated and properly tested, the system is fielded and begins production. This phase includes all of the processes needed to run the system in its operational environment. Some important security aspects include reoccurring user security training as well as user account maintenance and deletion.

Security Response Process

Operational systems still have flaws and a process to correct them should be in place. It may take several revisions before most system errors and security flaws are finally identified and fixed. Some security flaws may not show up for years until after system fielding (Beaver, 2006). Even with a fully certified and accredited information system, software that once was thought to be secure may not be as time passes. New vulnerabilities in the cyber world can introduce unanticipated security problems. For example, newer technologies such as small USB storage devices or a new Internet-based threat can introduce significant new security risks not anticipated in the initial risk assessment. Thus, an established organizational response process is essential for the rapid correction of newly identified errors and vulnerabilities. With a robust process in place, maintainers can quickly attempt to correct and test software flaws when new vulnerabilities surface. Once recertified, the corrected system can then be re-fielded.

Retirement

Some systems may still exhibit unique security risks even when entering the retirement phase. If an organization phases a system out, system administrators may relax security standards as management focuses on developing a replacement system. Yet, if the data in the retiring system is sensitive, administrators must continue to take due care in protecting data. A recent study of computer disks available on the second-hand market indicated that over 50% of the disks examined still contained significant volumes of sensitive information (Jones, Mee, Meyler, & Gooch, 2005). It remains critical that system administrators properly sanitize or dispose of sensitive data when an IS enters the retirement phase.

Dedicated Security Reviews and Audits

In addition to each of the seven phases in our notional model, a series of dedicated security reviews and audits can promote security integration throughout the SDLC. A security review can bring together the entire development team to focus on security. In addition, a security review should not just involve the security team, but all appropriate players on the project. It can be especially valuable to have the most experienced senior team members present at reviews so they can share their knowledge with the entire team. Significant gains can be realized by leveraging the experience of senior members and peers during reviews.

For lower risk systems, a dedicated review may not be necessary but security concerns should still be addressed during periodic system reviews. For higher risk systems, consideration should be given to hiring third-party professionals or consultants as they can provide a more objective review of the system under development (Howard, 2004). For example, an independent third-party team can specifically review the system security design before the implementation phase begins.

Security Considerations: Other Models of System Development

Other approaches to systems development exist that are variations, extensions, or streamlined versions of the notional “waterfall” model. Yet in each case, the same basic security challenge exists: developers can be tempted to push aside security requirements in the name of rapid development and thus fall prey to an “add security later” attitude. Following are a few security considerations for less traditional models of systems analysis and development.

Prototyping

Prototyping involves the rapid development of an early working system, often done using an iterative approach. Prototypes should include the security mechanisms appropriate for the prototype version. If developers delay putting security functionality until a later prototype version, they risk relegating security concerns to an afterthought.

Joint Application Development (JAD)

JAD is a team-based tool for gathering user requirements and creating system designs; it is often used during the earlier stages of the SDLC. With this team approach, it may be advantageous to have at least one member who understands and can articulate the security needs of the proposed system.

Rapid Application Development (RAD)

RAD is similar to a condensed version of the SDLC. RAD also uses a team approach involving users, managers, and IT professionals. The RAD methodology compresses analysis, design, implementation, and test phases into a series of shorter, iterative development cycles. With its stress on rapid, iterative development, security considerations and requirements can get lost without proper security representation on the team.

Extreme Programming (XP)

This is a team-based approach based on rapid prototyping; XP uses an evolving design, daily testing, and integration to design the system. Rather than using longer “traditional” development cycles, XP attempts to blend all SDLC phases, a little at a time, throughout a software development project (Beck, 1999). However, an evolving requirements approach does not necessarily ensure security considerations will be properly implemented into the system. Again, programmers must take care to ensure that security needs are included using this rapid approach.

Systems Assembly from Reusable Components

This process involves more of an assembly of a system from existing parts rather than the creation of a new system. It can be complex because developers need to validate that the individual components are secure before assembling the system. Insecure code with flaws can easily be placed in a reusable software library. Thus, developers should not simply copy code and assume the code is secure.

FUTURE TRENDS

Several key trends that affect the development of secure systems are worthy of mention. Following, three are briefly discussed: outsourcing, litigation, and automated tools.

Outsourcing

In many environments, increased competition has necessitated the hiring of outsourced workers to develop lower cost software. In some instances, requirements analysis and design phases are conducted in a home country while coding-intensive work is outsourced to third parties often located in foreign countries. This trend can make developing secure systems more challenging since outside players are involved in the software development team. Management should carefully evaluate the added risks of outsourcing high-security systems to third parties. If a software development project represents a high-risk to a company, hiring third parties from a foreign country where different laws apply could add unnecessary risk compared to hiring in-country sources.

Litigation

A number of recent legal mandates worldwide have affected the information security field including the Sarbanes-Oxley Act, Gramm-Leach-Bliley Act, California Senate Bill 1386, European Union Data Protection Directive, and the Basel II Accord. The trend is toward greater corporate accountability for security breaches. If insecure software is partially the reason for a costly security incident, the developers of the system may be liable for damages. Thus, due diligence must be exercised when developing systems that process sensitive data (e.g., privileged information), that supports human life (e.g., medical systems), or that protects expensive resources (e.g., climate control software).

Automated Tools

Today, tools are available to support every phase of the SDLC and are often part of a computer aided software engineering (CASE) tool suite. Tools are available for source code management, requirements analysis, requirements traceability, defect and correction tracking, test management, as well as production phases. Not all tools are the same and developers should carefully choose tools to meet the specific needs of a project. If a tool is inappropriate for the task, it can be detrimental to the effort or simply not used by the development team. Fortunately, today one can find a number of available automated tools specifically designed to support the security needs encountered during software development (e.g., Giorgini, Massacci, Mylopoulos, & Zannone, 2005).

CONCLUSION

Considering the importance of cyber security in the modern world, the need to develop secure information systems is critical. In this chapter, emphasis was placed on identifying security risks and documenting requirements as early as possible. In short, developers should document initial requirements and security considerations should remain a priority in each phase of the SDLC. In addition, attention was given toward improving security with automated tools, performing abuse cases, tracing security requirements, holding regular security reviews, conducting certification and accreditation, and developing security response processes. By following these steps, practitioners will be able to improve both the security and the overall quality of computerized information systems.

RESOURCES

Resources are available to guide the development of secure systems. Following is a list of useful documents that can aid information system developers.

1. Systems Security Engineering—Capability Maturity Model (SSE-CMM). Site: http://www.sse-cmm.org
2. NIST Special Publication 800-64—Security Considerations in the Information System Development Life Cycle. Site: http://csrc.nist.gov/publications/nistpubs/
3. NIST Special Publication 800-37—Guide to the Certification and Accreditation of Federal Information Systems. Site: http://csrc.nist.gov/publications/nistpubs/
4. ISO/IEC 17799:2005—Code of Practice for Information Security Management. Site: http://www.iso.org.
5. Microsoft's Security Developer Center and the Trustworthy Computing Security Development Lifecycle. Sites: http://msdn2.microsoft.com/en-us/security or http://msdn.microsoft.com/security/

NOTE

Opinions, conclusions, and recommendations expressed or implied within are solely those of the authors and do not necessarily represent the views of USAF Academy, USAF, the DoD, or any other government agency.

REFERENCES

Ayer, S., & Patrinostro, F. (1992). *Documenting the software development process*. New York: McGraw Hill.

Beaver, K. (2006). *Securing your software development life cycle*. Security Park.net. Retrieved July 28, 2006, from http://www.securitypark.co.uk/article.asp?articleid=25356

Beck, K. (1999). Embracing change with extreme programming. *IEEE Computer, 32*(10), 70-77.

Berg, R. (2006). *Implementing source code vulnerability testing in the software development cycle* (White Paper). Waltham, MA: Ounce Labs.

Bishop, M. (2005). *Introduction to computer security*. New York: Addison-Wesley.

Flechais, I., Sasse, M. A., & Hailes, S. M. V. (2003). *Bringing security home: A process of developing secure and usable systems.* Paper presented at the New Security Paradigms—ACM Workshop, Ascona, Switzerland.

Geer, D. (2002). *Information security: What the markets want and why?* Dartmouth College Institute for Security Technology Studies. Retrieved August 1, 2006, from http://www.ists.dartmouth.edu/speaker_series/geerslides.pdf

Geer, D., Hoo, K. S., Jr., & Jaquity, A. (2003). Information security: Why the future belongs to the quants. *IEEE Security & Privacy, 1*(4), 24-32.

Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2005, September). *St-tool: A CASE tool for security requirements engineering.* Paper presented at the 13th IEEE International Conference on Requirements Engineering.

Hansche, S., Berti, J., & Hare, C. (2004). *Official (ISC)^2 guide to the CISSP exam*. New York: Auerbach.

Howard, M. (2004). Building more secure software with improved development processes. *IEEE Security & Privacy, 2*(6), 63-65.

Jones, A., Mee, V., Meyler, C., & Gooch, J. (2005). Analysis of data recovered from computer disks released for sale by organizations. *Journal of Information Warfare, 4*(2), 45-53.

Jordan, M. (2006). *Tools for securing the software development lifecycle (webcast presentation)*. Search Security. Retrieved July 27, 2006, from www.searchsecurity.com

Knapp, K. J., Marshall, T. E., Rainer, R. K., Jr., & Morrow, D. W. (2004). *Top ranked information security issues: The 2004 international information systems security certification consortium (ISC)^2 survey results.* Alabama: Auburn University.

Murray, W. H. (2001). Common system design flaws and security issues. In H. F. Tipton & M. Krause (Eds.), *Info. security management handbook* (Vol. 2, pp. 291-303). Boca Raton, FL: CRC Press.

Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering, 27*(1), 58-93.

Royce, W. W. (1970, August). *Managing the development of large software systems.* Paper presented at the IEEE WESCON.

van Wyk, K. R., & McGraw, G. (2005). Bridging the gap between software development and information security. *IEEE Security & Privacy, 3*(5), 75-79.

Whitten, J. L., Bentley, L. D., & Dittman, K. C. (2000). *Systems analysis and design methods* (5th ed.). New York: McGraw-Hill Irwin.

KEY TERMS

Abuse Cases: Special security requirements can be documented in *abuse cases* where the system developers document and test ways that the system can be maliciously used and how it could affect the system. When fielded, the system can then be tested against these abuse cases to see if the system is adequately protected.

Certification and Accreditation (C&A): Completed systems can go through a process that evaluates the security stance of the system against a predetermined set of security standards while also testing how well the system performs its intended functional requirements. After system certification, management then accredits a system, accepts responsibility, and is accountable for adverse impacts.

Extreme Programming (XP): A team-based development approach based on rapid prototyping that uses an evolving design of often daily testing and integration to design the system. Rather than using longer development cycles, XP advocates blending all SDLC phases, a little at a time, throughout a software development project.

Initial Systems Risk Assessment: During the investigation phase of the systems development life cycle, analysts investigate the required security level of a proposed information system. This initial risk assessment should be grounded in the business needs of the system and can be used as a basis for future security decisions.

Joint Application Development (JAD): Introduced by IBM in the 1970s, JAD is an interactive, team-based approach to systems development that uses groups of individuals for collecting user requirements and creating system designs.

Rapid Application Development (RAD): RAD offers a condensed version of the entire development process while emphasizing the use of prototypes and automated tools to help speed up application development. RAD also uses a team approach composed of users, managers, and IT professionals to help complete the project.

Response Process: Once fielded, software that once was thought and even accredited to be secure may not be as secure as time passes. New threats and vulnerabilities in the cyber world can introduce unanticipated security problems. An established organizational response process can help the security team quickly address newly identified threats and vulnerabilities. The corrected system should be re-tested before re-fielded.

Security Requirements Traceability: Requirements traceability is intended to ensure continued alignment between security requirements and various outputs of systems development. Implementing security requirements traceability can help designers identify areas in a developing system that do not meet the documented requirements.

Systems Development Life Cycle (SDLC): The SDLC is a systems approach to developing information systems and is notionally made up of several phases: investigation, requirements analysis, design, implementation, integration, operations, maintenance, and retirement.

System Security Requirements Document: A stand-alone requirements document that gathers the security requirements of a proposed system.

ENDNOTES

1 For details of the research study, see Knapp, Marshall, Rainer, and Morrow (2004).

2 For an expanded discussion of managing the development of software systems, see Royce (1970).