Information Systems Security

ISSN: 1065-898X (Print) 1934-869X (Online) Journal homepage: https://www.tandfonline.com/loi/uiss19

Secure Coding: Building Security into the Software Development Life Cycle

Russell L. Jones & Abhinav Rastogi

To cite this article: Russell L. Jones & Abhinav Rastogi (2004) Secure Coding: Building Security into the Software Development Life Cycle, Information Systems Security, 13:5, 29-39, DOI: 10.1201/1086/44797.13.5.20041101/84907.5

To link to this article: https://doi.org/10.1201/1086/44797.13.5.20041101/84907.5

Secure Coding: Building Security into the Software Development Life Cycle

Russell L. Jones and Abhinav Rastogi

Many of the security properties that are outlined repeatedly in the newer regulations and standards can easily be side-stepped. Too often the culprits are unsophisticated software development techniques, a lack of security-focused quality assurance, and scarce security training for software developers, software architects, and project managers. To meet future needs, opportunities, and threats associated with information security, security needs to be "baked in" to the overall systems development life-cycle process.

Information security and privacy loom ever larger as issues for public and private sector organizations alike today. Government regulations and industry standards attempt to address these issues. Computer hardware and software providers invest in meeting both regulatory and market demands for information security and privacy. And individual organizations — corporations and government agencies alike — are voicing concern about the problem.

AN URGENT AND GROWING NEED

With increasing globalization, extended enterprises, "e-motivated" consumers, and the ubiquitous technologies that make it all possible, the need for information security is taking on greater urgency than ever before.

RUSSELL L. JONES, CPA, CISA, CISSP, CISM, is a Principal in Deloitte's Security Services Group. He specializes in integrating third-party security products with complex e-business and ERP applications and related infrastructures. With more than 13 years of experience, he has a strong focus on the design, architecture, implementation and deployment of Identity Management solutions (i.e. Web Access Management, User Provisioning), encryption solutions, and distributed architecture application solutions. Russell also has significant experience with the development of security architectures, network security analysis and design, and role-based access control (RBAC) design and deployment. Russell has practical experience applying security frameworks such as ISO17799 and ISO15408 against real world environments. Russell is a key member of Deloitte's Identity Management Council.

ABHINAV RASTOGI, CISSP, is a Sun-certified Java engineer with more than seven years of experience in designing and developing enterprise-level applications. He has extensive training and experience in installation, configuration, and integration of Identity Management, Single Sign-On, and Web Security products. He has experience in Java Security framework in, JCE, and JSSE, as well as in Server-side Java Technologies such as EJBs, Java Servlets, JavaServerPages, JDBC, Java Beans, and XML. Abhinav also has sound knowledge of Web Protocols and Web Technologies including TCP/IP, HTTP/HTTPS, SSL, JavaScript, and DHTML.

Public and private sector organizations must guard their intellectual properties and intellectual capital. Consumers need to protect their private information. And organizations that gather private information from consumers for whatever purpose should be safeguarding that data through every phase of its use. Yet the increasingly pervasive sharing of information along data networks and between computer applications presents a seemingly insurmountable challenge.

Governments around the world are reacting with legislation and regulations that seek to establish requirements for information security, either explicitly or implicitly. The Gramm–Leach–Bliley Act (GLBA), the Health Insurance Portability and Accountability Act (HIPAA), CFR 21 Part 11, and the EU Data Protection Directive, for example, outline specific security and privacy requirements that share common themes.

The Sarbanes–Oxley Act of 2002, on the other hand, deals with effective internal controls within public corporations. Although not prescriptive with respect to security, it does outline that both general controls and application-level security controls are required.

Other efforts are underway worldwide to rein in issues involving information security and privacy, including:

- ☐ IBM's Privacy Management Council and Privacy Management Institute
- ☐ Microsoft's Trustworthy Computing Initiative
- ☐ Liberty Alliance Project
- ☐ Watchfire's PrivacyXM

Private and public organizations are also attempting to establish global standards for information security and privacy. These include:

- ☐ Oasis, the Organization for the Advancement of Structured Information Security
- ☐ The Open Standards Board in the EU
- ☐ P3P (Platform for Privacy Preferences), developed by the World Wide Web Consortium
- ☐ Seal programs, such as American Institute of Certified Public Accountants' WebTrust seal and BBBOnline
- ☐ ISO 17799, published by the International Standards Organization
- ☐ The IT Infrastructure Library; ITIL is a series of documents that are used to aid the implementation of a framework for IT Service Management (ITSM)
- ☐ National Institute of Standards and Technology (NIST)

These actions do not directly address the critical disconnect that is at the root of most security and privacy breaches: the failure of software developers to take a security view of their products from inception through deployment and beyond.

TABLE 1 Weighing the Benefits and Risks

The changing regulatory environment is already having a major effect on organizations' approach to security. New regulations, such as Section 404 of Sarbanes–Oxley, compel senior executives of publicly traded companies to be accountable for the effectiveness of their organization's internal control environment. Consequently, more executives are asking, "Are we secure?"
Chances are that if security is not built into their SDLC, effectively implemented, and continually monitored and refined, the answer to the question is most likely "no."
Organizations that aggressively pursue compliance with Sarbanes–Oxley and other recent regulations have a unique opportunity to address security. Unlike Y2K, this is not a one-time event. Sarbanes–Oxley is about changing process and behavior to ensure that organizations fairly represent their financial statements.
As a result, this is a perfect time to ensure the institutionalization of security. Organizations that undertake this process will enjoy material advantages in terms of regulatory compliance, true information security, and competitiveness. Competitiveness?
Companies can leverage the confidence that comes with baking security into the SDLC through the brand image they portray to their markets.
On the other hand, companies that do not rethink security as a part of doing business and begin the process of building it into the SDLC put much at risk. They likely will, in the end, spend much more time, money, and resources addressing problems that stem from security breaches and legal and regulatory noncompliance.

COMPLEXITY AND CHANGE COMPLICATE THE PROBLEM

The ability to implement the types of security controls described above has become much more complex and challenging in the past decade. The rapidly evolving nature of the technology and information systems that underlie most major public and private sector organizations is at the root of the problem.

The Internet and the technologies that it has spawned have opened up limitless possibilities for commerce via Web-based supply chains. On the other hand, the traditional boundaries around organizations are quickly becoming blurred because of those same technologies. Traditional firewall systems have become less effective in preventing or detecting Web-based attacks.

In fact, at the heart of many types of successful system attacks that are seen today are poorly developed systems and applications. Many of the security properties that are repeatedly outlined in the regulations described above, including accountability, unique user accounts, and confidentiality, can be circumvented when software developers have not paid enough attention to security in the design, development, deployment, and maintenance of their products.

A SURPRISINGLY PERVASIVE ISSUE

Over several years, we have been involved with many security consulting engagements. Among these were projects entailing:

- ☐ Network and application-level penetration testing
- ☐ Vulnerability assessments
- ☐ Enterprise security architecture
- ☐ Identity management strategy, architecture, and implementation
- ☐ Distributed system architecture design, build, and testing
- ☐ Custom encryption development for client/server-based transaction systems

In every engagement, we discovered — and, in several cases, successfully exploited — various types of security vulnerabilities. Sometimes we discovered the vulnerabilities tangential to the process of meeting the core objectives of the engagement. In other instances, such as penetration testing, the actual objectives of the engagement were to discover and exploit system vulnerabilities.

Regardless, the root cause behind the vulnerabilities ended up being the same: While many of these organizations had formal enterprisewide systems development life-cycle (SDLC) processes, none had baked security into those processes.

TABLE 2 A Case of Institutional Indifference

A large insurance company engaged us to implement a Web single sign-on solution (WSS). The implementation required upfront analysis of six Web applications that the WSS would support, along with analysis of the network environment, security policies, procedures, and standards.

Our analysis of the applications revealed totally distinct and different approaches to identity authentication. This posed a problem for the WSS, which required a number of custom-developed components. These components would map the authentication credential generated by the WSS into an authentication credential understood by each of the six applications.

This problem, as we discovered, was only the symptom. The root cause was the organization's "official" SDLC process. Yes, the company had an established SDLC process. But it was, effectively, shelfware; none of the developers, system architects, or project managers used it. Worse yet, some of their people used no formal SDLC at all. They simply "winged it." Other project teams used established software development processes. But they did not consistently document design and code artifacts.

When we met with different project leads and their developers, it was clear that security concerns were not part of their thought processes. Nor had they ever considered that security should be implemented from the initial requirements phase. In fact, they consistently expressed that the company's security group addressed security.

Yet when we met with the security group, we discovered that their primary focus was securing the network through DMZ design, firewall setup and configuration, and virtual private networks, among others. This indifference toward application security put this organization at significant risk of intrusion from within or outside the organization.

The reality is that information security is an afterthought for many organizations. So most vendors only give it a token nod. Most often, security is not an integral part of their business or information strategies. Nor is it woven into their IT projects.

In fact, they usually wait until the system has been designed, tested, and is ready for deployment. This approach is analogous to designing and prototyping a car, building it, roadtesting it, and then determining a few months before rolling out production models that it needs brakes.

The reasons for this neglect vary.

- ☐ Security is not considered a business enabler or revenue generator. Its value is not appreciated until risks manifest themselves through attacks and compromises; by then it's too late.
- ☐ In the absence of proper support from the top management, security flounders and has few champions in the organization.
- ☐ Developers are usually hired for their development or coding expertise and may have minimal security knowledge. Nor are they given the training, tools, resources, time, or, in many cases, the motivation to build secure systems.
- ☐ Software projects inevitably run up against deadlines, wherein companies focus their primary effort on the software's features rather than its security.
- ☐ Security is often perceived as a barrier to functionality, adding constraints and reducing flexibility.
- ☐ Security is often treated as a collection of tasks that can be completed at the end of the project rather than as a process or a mindset employed throughout the project.

The cost of these lapses, regardless of the reasons behind them, can be high. Microsoft Corporation and, by default, many thousands of organizations and individuals experienced the effects of the problem in 2003 when security holes in Microsoft products were exploited, resulting in billions of dollars in recovery costs and lost productivity. The company launched a massive, good-faith attempt to remedy the problem, yet the problems persist today.

GAINING WIDER RECOGNITION

If the security considerations for systems were woven into the SDLC and the developers, project managers, and system architects were given adequate security training, many of the security vulnerabilities that manifest themselves in production systems — buffer overflows are a common example — would never appear.

The National Institute of Standards and Technology (NIST) recently released a special publication called "Security Considerations in the Information System Development Life Cycle."[1] This document was developed primarily for federal agencies that are grappling with how to develop secure systems, as well as how to articulate to vendors what they need with respect to secure software products. The NIST document offers a framework for incorporating security considerations throughout a generic SDLC, including a "minimum" set of security considerations that need to be considered within each of the five phases that they present.

TABLE 3 Secure System Design and Deployment

Traditionally reference materials on the art of secure system design and deployment have been scarce. Thankfully, this is beginning to change. Three seminal works are:

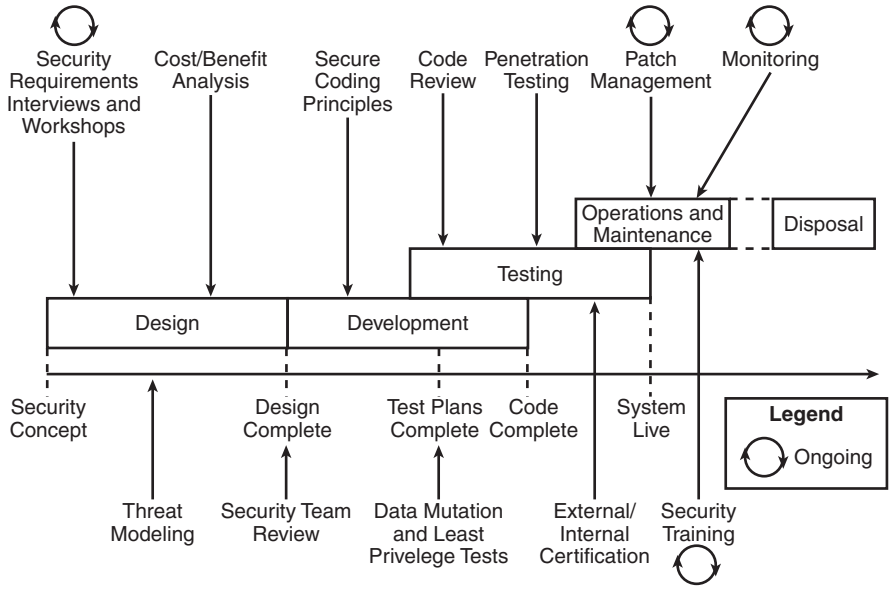
- ☐ *Attack Trees* by Bruce Schneider
- ☐ *Building Secure Software* by John Viega and Gary McGraw
- ☐ *Writing Secure Code, 2nd ed.,* by Michael Howard and David C. LeBlanc

Other valuable sources of information include:

- ☐ *Systems Security Engineering Capability Maturity Model* (SSE-CMM), provided by the International Systems Security Engineering Association (ISSEA): www.sse-cmm.org
- ☐ *Common Criteria for Information Technology Security Evaluation*, provided by the National Institute of Standards and Technology: www.nist.gov

These and other sources offer extensive details regarding specific programming techniques and risk frameworks.

FIGURE 1 Secure Software Development Life Cycle



In addition to NIST, a number of leading IT research groups have recently recognized the need for security to be integrated into the SDLC. Meta Group Research, for example, released a three-part research paper in 2003 entitled "Application Securability." Of particular interest is Part 3 of this series, entitled "Secure Application Development Principles."[2]

THE SOLUTION: BAKED-IN SECURITY

Starting with an understanding of business objectives and security requirements, our software development methodology ensures that security is designed into the software and thoroughly tested and measured throughout the development life cycle. This not only decreases the risk of applications being compromised after a perimeter security breach but also ensures protection against infrastructure failures and insider attacks.

We have taken a detailed look at the software development life cycle (see Figure 1) and interwoven security in its every stage. Our methodology is based on existing risk management and a typical software building life cycle.

DESIGN PHASE

Security Context

The life cycle has begun. The first and foremost security task is to involve a qualified security person in the project as the security lead. The security lead proceeds with forming the security team and identifies and involves the key stakeholders of the project.

The security team and the key stakeholders then determine the security context of the system by capturing a high-level security-oriented view of the system operation. It is important to understand the ultimate goal of the development, and what level of access or exposure is ultimately being requested.

Requirements Gathering

Gathering the correct set of applicable security requirements is very important, as they are the main input to the security architecture design. This comprises the following steps:

- ☐ Conducting workshops and interviews to gain an understanding of the business owner's security requirements
- ☐ Ensuring that the organizations' security standards are included

Residual risk tables can help an organization look at risks, threats, mitigating controls, and residual risk levels once the mitigating control is in place.

- ☐ Rationalizing and enhancing the security requirements from industry best practices and standards such as ISO 17799 and ISO 15408
- ☐ Identifying and including the regulatory privacy and security requirements, laws, policies, standards, external influences, and constraints such as Gramm–Leach–Bliley (GLB) that govern the application

Risk Assessment

As the design begins to take shape, the security team performs the risk assessment, which is the process of analyzing and measuring risk, and establishing an acceptable level of risk for the system. Risks are assessed by examining possible threats, identifying the vulnerabilities, and considering the impact if the identified vulnerabilities were exploited by the threats. Once the risk(s) are identified, a risk mitigation strategy can be established and a cost-benefit analysis can be conducted to determine the most cost-effective way to mitigate the risks.

Asset Identification and Valuation. Important to any risk assessment is the identification of assets, both tangible and intangible, their replacement costs, and the valuation of information asset availability, integrity, and confidentiality. These values may be expressed in monetary or nonmonetary terms for quantitative or qualitative purposes, respectively. This step is important for the final cost-benefit analysis of security controls.

Threat Modeling. Threat modeling is the core step in any security solution. It is a way to start making sense of the "vulnerability landscape." What are the real threats against the system? If you don't know that, how do you know what kind of countermeasures to employ?

Threat modeling should include the following:

- ☐ Create application model: e.g., data flow diagrams and UML diagrams
- ☐ Identify attack target nodes: identify the nodes in the application model that can become a target for a security attack. The attack nodes usually have the following characteristics:
  - – Greater surface attack area (e.g., Web server in a DMZ).
  - – Carrier of sensitive information (e.g., database storing credit card information).
- ☐ Identifying the attack target nodes in an application model allows the security team to critically analyze all parts of the application at the component level.
- ☐ Categorize threats to each attack target node (STRIDE):
  - – Spoofing: Is the target node susceptible to spoofing?
  - – Tampering: Can the target node be tampered with?
  - – Repudiation: Can an attacker repudiate this action?
  - – Information disclosure: Can an attacker view this item?
  - – Denial of service: Can an attacker deny service to this process or data flow?
  - – Elevation of privilege: Can attackers elevate their privilege by attacking this process?
- ☐ Building a threat tree that depicts all possible vulnerabilities (see Figure 2). The purpose of a threat tree is to visually describe the areas of the system vulnerable to specific threats. Each threat can further comprise multiple subthreats.
- ☐ Evaluating threats on the basis of the following:
  - – Damage potential.
  - – Reproducibility.
  - – Exploitability.
  - – Affected users.
  - – Discoverability.

By assigning the above parameters to each threat, we create a more realistic vulnerability landscape. This subsequently helps in the risk mitigation strategy, when security controls and mechanisms are designed.

FIGURE 2 Threat Tree

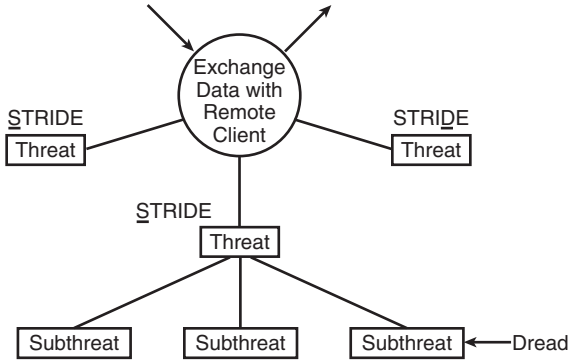
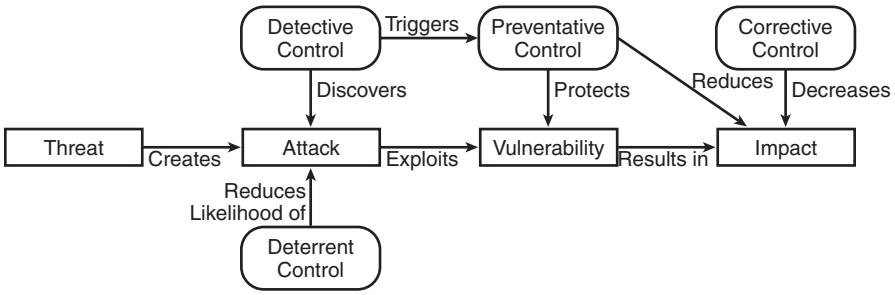


FIGURE 3 Security Controls



Risk Mitigation. Once threat modeling is completed for the application, the next step is to determine how to reduce risks to an acceptable level. Residual risk tables can help an organization look at risks, threats, mitigating controls, and residual risk levels once the mitigating control is in place.

Each threat identified in the threat tree is analyzed as to how it can give rise to an attack and how a vulnerability can be exploited. Adequate security controls are put in place to mitigate the risk presented by each threat.

These security controls (see Figure 3) can be of four categories:

1. Corrective Control
2. Deterrent Control
3. Detective Control
4. Preventive Control

Finally, residual risk tables (see Table 4) are created to summarize and quantify the results of threat modeling and risk mitigation. Residual risk tables further include the valuable cost-benefit analysis of different possible security controls for each threat and corresponding savings per year.

After the creation of the risk residual table, all different possible security controls are analyzed and weighed against each other. Depending on the risk mitigation strategy and taking into account the overall security budget the appropriate security controls are selected.

System Design

In the system design phase, the security controls are given the shape of high-level system design components. The system design provides a clear understanding of

TABLE 4 Residual Risk Table

	Asset		Threat Severity	Incidents			Security Control/ Mechanism		Savings/ Year
	Name	Value		Cost/ Incident	Incidents/ Year	Cost/ Year	Name	Cost	
Attack Target Node									
Threat 1									
Threat 2									

how security is provided by the security controls, how trust is managed, and how the system would withstand the major attack paths. Major technologies used (e.g., the system uses authentication and delegation through Kerberos) and security protocols (e.g., SSL and SSH) are represented in the system design.

The purpose of system design is to verify that:

- ☐ The system design should be compliant with the enterprise security architecture.
- ☐ All security requirements are interpreted and turned into actual security design components.
- ☐ The system design includes the following security principles into the software:
  - Securing the weakest link.
  - Practice defense in depth.
  - Compartmentalize.
  - Keep it simple.
  - Promote privacy.

If a COTS product is part of the solution then the system design phase can also include the product/vendor selection process.

Security Review. Once completed, the system design is thoroughly reviewed by the security team. The review ensures that all security requirements are met and risk is mitigated to acceptable levels.

DEVELOPMENT PHASE

This is the phase in which all software components are coded by the developers. It is at this stage where vulnerabilities such as buffer overflow inadvertently find their ways into the software. Buffer overflow is by itself the cause of more than 50 percent of security compromises. It can't be stressed more how important it is to safeguard the code against letting in such vulnerabilities. How is this achieved? It is achieved by ensuring that the developers are

- ☐ Cognizant of security risks
- ☐ Adept in secure coding

Cognizance of Security Risks

It is the job of the information security office and project leadership to stress the importance of security to the whole team. They should know the security policy, standards, guidelines, and procedures of the organization. The developers should understand the relevance of incorporating security and should not perceive it as a hindrance.

Secure Coding

Every developer needs to be adept in the basic tenets of coding secure applications. These tenets should always be in the minds of the developers and should be reflected in the code that they write. These fundamental security guidelines and rules that need to be followed when writing code are summarized below.

- ☐ *Input validation*: Always validate all input to the system. The input can be user input, environment variables, file input, and so on.

- ☐ *Hostile environment*: Always assume a hostile environment externally as well as internally.
- ☐ *Open standards*: Open standards have passed the scrutiny of the community and are more reliable.
- ☐ *Trusted components*: Having components that trust one another is always risky. Minimize and protect the trusted components in the system.
- ☐ *Data in process, transit and storage*: Have safeguards to protect the data wherever it exists.
- ☐ *Always authenticate*: Never allow any access, user or programmatic, without proper authentication.
- ☐ *Native security protections*: Do not subvert the native or in-house security.
- ☐ *Fail securely*: Ensure that if an application fails it is not less secure.
- ☐ *Log, monitor, and audit*: Always log and monitor what's going on in the system. Logs are the best detective security controls.
- ☐ *Date and time*: Use accurate system date and time.
- ☐ *Least privilege*: Provide least privilege and limit access to need to know.
- ☐ *Exception handling*: Exceptions should be handled at the appropriate level in the code and provide sufficient information about the exception.
- ☐ *Strong cryptography*: Strong ciphers and algorithms that are industry standards should be used. Stay away from homegrown or proprietary cryptography solutions.
- ☐ *Random numbers*: The random numbers should be nondeterministic.

Security Safeguards

From a coding perspective there are a number of vulnerabilities that seem to reoccur that weaken the security of a solution. Developers should be proactive in guarding against them.

- ☐ Inherent vulnerabilities of the programming language
- ☐ Security through obscurity
- ☐ Buffer overflow and format string vulnerabilities
- ☐ Race conditions
- ☐ Locking problems
- ☐ Trusting user input and not validating parameters
- ☐ Poor exception handling
- ☐ Complexity
- ☐ Control granularity
- ☐ Excessive privilege

TESTING PHASE

There are two aspects regarding security testing. One of them is the testing performed by the testing or QA team. The other is the formal evaluation required for certifications.

For security testing, the security team works with the developers and the test team to develop a security test plan, which is directed by system risks that were unearthed during the risk assessment phase. Security testing ensures that countermeasures are correctly implemented and that code is developed following security best practices. There are four security activities in this phase.

Unit Testing

As system modules are developed, the security team defines tests, such as data mutation and least privilege tests, for each module to test its security behavior.

Specifically, all buffers for a module are checked to determine if they are susceptible to overflows. This is done through data mutation tests that are executed by injecting environment faults to the application. For doing this, the environment faults are divided into user input, environment variable, file system input, network input, and process input.

The source-code analysis tools (RATS, ITS4, Splint, MOPS, etc.) and runtime testing methods, for example, the fault injection systems, can also be used to check for the presence of flaws.

If resources permit, it is good practice to have independent code reviews at this stage. However, for many large organizations it is impractical to review all code,

Specific tests should be in place to check authentication, authorizations, and entitlements, and to test all countermeasures that were put in place as a result of the risk analysis.

but the security team may want to require that certain code with major security implications (e.g., code that performs authentication) undergoes review.

Integration/Quality Assurance Testing

This is the stage where the system modules are being assembled and tested as an integrated application. The security team should ensure that security testing is embedded in the overall test plan for the product at this stage. Specific tests should be in place to check authentication, authorizations, and entitlements, and to test all countermeasures that were put in place as a result of the risk analysis.

Penetration Testing

The last stage of testing before the site is entered into production is to conduct a full penetration, or ethical hacking, test. This final level of security testing involves professional hackers attempting to penetrate the system. The testing should be done from the perspective of an outsider with no approved system access and a malicious insider who has access to the system. It is often a good idea to have this testing done by a third party who had no involvement in the development or design of the system.

Certification

In more regulated environments systems are required to be security-certified. The meaning of such certification is specific to each evaluation program. Some of the certifications assess the security capabilities of a system, such as whether it has authentication or authorization services and not whether it cannot be hacked. Examples of such software certifications are FIPS and Common Criteria.

OPERATIONS AND MAINTENANCE PHASE

The system is finally live and is being used by actual users. The system has to be diligently maintained. The following security tasks are necessary in this stage:

- ☐ Managing patches and updates
- ☐ Conducting security training
- ☐ Managing cryptographic material — keys, certificates, and so on
- ☐ Maintaining user administration and access privileges
- ☐ Documenting all security-relevant information.
- ☐ Defining response process for handling security bugs
- ☐ Defining backup procedures
- ☐ Defining business continuity procedures
- ☐ Monitoring of logs and alerts
- ☐ Ensuring that audit logs are available
- ☐ Reviewing the physical protection
- ☐ Reviewing offsite storage usage, services, and availability
- ☐ Running the operation assurance activities such as reviewing actions of people who operate/run the system (e.g., change control procedures, log review procedures, etc.)

DISPOSAL PHASE

Disposal is the phase concerned with resolving the disposition of the information, software and hardware, whether it involves moving, sanitizing, disposing, or archiving. Security is a concern not only prior to and during the system's life, but also when disaster occurs and during disposal.

Security-related activities during this phase should include:

- ☐ For encrypted data, ensure long-term storage of cryptographic keys.
- ☐ Review the legal requirements for records retention — e.g., Freedom of Information Act requirements.
- ☐ Sanitize the media — for example, through overwriting, de-Gaussing, or destroying the data on the storage media.

INFORMATION SECURITY: A PROCESS AND A PHILOSOPHY

There will always be opportunities for improvement in the area of information security. However, the most effective way to approach security is as an iterative and evolutionary process.

Consider the following as you begin the process of integrating security into your SDLC process:

- ☐ Gain support from the highest levels of your organization. This means the "C-suite," that is, the CEO, CFO, CIO, and so on. For companies that are publicly traded and have to comply with Section 404 of Sarbanes–Oxley, security is getting the most visibility that it has ever received at the C-suite level. Now is a good time to leverage that visibility to the benefit of the entire enterprise.
- ☐ Assess your organization's current capability with respect to secure coding. You may find that you already have relatively effective processes, procedures, and methodologies in place. These should be incorporated into the overall, enterprise-wide, secure SDLC process.
- ☐ Develop a phased approach for integrating security into your SDLC. Change management techniques will need to be included in the process if you are to gain the support and acceptance of the process among developers, project managers, and system architects. Again, top-level support within your organization will assist in this process.
- ☐ Make sure your developers, project managers, and system architects undertake formal training with respect to secure coding. A number of training organizations are beginning to offer this type of training, including Foundstone, Global Knowledge, and Learning Tree.
- ☐ Be sure that secure coding reference materials are available to your developers, project managers, and system architects, including those mentioned in this article. Each of them offers useful tools, techniques, and approaches.
- ☐ Establish internal metrics and key performance indicators that can be used to determine the progress and success of your security evolution.

Weaving security into the fabric of your organization's IT activities in this way will pay handsome dividends, both near term and over time.

Notes

1. Tim Grance, Joan Hash, and Marc Stevens, "Security Consideration in the Information System Development Life Cycle," NIST, October 2003.
2. Perkins, Earl, "Application Securability," Meta-Group Research, February 2003.