The Ten Best Practices for Secure Software Development

Mano Paul, CSSLP, CISSP, AMBCI, MCAD, MCSD, Network+, ECSA

Introduction

Building secure software is the responsibility of all the stakeholders involved with the software development lifecycle (SDLC). While the security of software can be attributed to the technologies chosen or processes followed, eventual accountability is ascribed to the people building it. Inherently secure technologies are limited and in cases when chosen, the likelihood that they are implemented securely is isolated. Many times, the processes that are in place to aid in the security of software end up being circumvented, victims of the iron triangle of project scope, schedule, and budget.

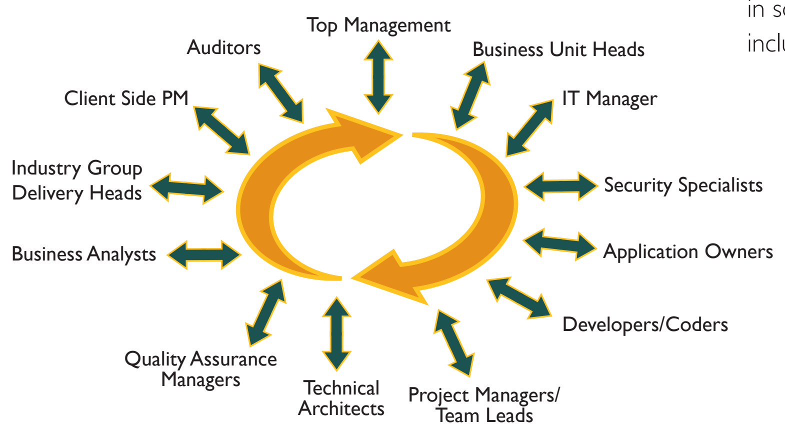
(ISC)$^{2®}$'s mantras, "It's the People" and "Security Transcends Technology" best epitomizes the fact, that the "people" component of security is crucial in building secure software.

(ISC)^{2}'s whitepapers, *The Need for Secure Software, Software Assurance: A Kaleidoscope of Perspectives*, and *Software Security: Being Secure in an Insecure World*, address the "Why", "What" and "How-Tos" of designing, developing, and deploying secure software. This whitepaper focuses on the human element – the "*Who*" and will center around "The Ten Best Practices" that a secure software lifecycle professional should follow to build secure, hack-resilient, and compliant software.

"In the 80's we wired the world with cables and in the 90's we wired the world with computer networks. Today we are wiring the world with applications (software). Having a skilled professional capable of designing, developing and deploying secure software is now critical to this evolving world."

Mark Curphey,
Director & Product Unit Manager, Microsoft Corporation,
Founder of Open Web Application Security Project (OWASP)

Figure 1. Stakeholders in the SDLC



The Ten Best Practices

Software development involves many stakeholders, as depicted in Figure 1[a]. They can range from the analyst (business/requirements), to architects, coders, testers, and operations personnel. Development can also include management (product/project/personnel), and in some cases even executive-level management. Additionally included may be members from the security and audit teams.

A secure software lifecycle professional (SSLP) is any stakeholder who is responsible for building software with the goal of ensuring that the software built is not susceptible to security breaches. It must be understood that no software is 100% secure. However, software can be designed, developed, and deployed with a secure mindset, factoring in necessary security controls that minimize the likelihood of exposure and the impact if exploited. The following practices can help fulfill the SSLP's mission of building hack-resilient software.

The Ten Best Practices

1. **Protect the Brand Your Customers Trust**
2. **Know Your Business and Support it with Secure Solutions**
3. **Understand the Technology of the Software**
4. **Ensure Compliance to Governance, Regulations, and Privacy**
5. **Know the Basic Tenets of Software Security**
6. **Ensure the Protection of Sensitive Information**
7. **Design Software with Secure Features**
8. **Develop Software with Secure Features**
9. **Deploy Software with Secure Features**
10. **Educate Yourself and Others on How to Build Secure Software**

Best Practice #1: Protect the Brand Your Customers Trust

The Harvard Business Review special publication, "Breakthrough Ideas for 2008,"[b] listed "Cybercrime Service Economy" as one of the top 20 transformations of the business world. Scott Berinato, Executive editor of *CSO magazine*, who contributed to the publication, asserts that the new breed of hackers don't just cause interruptions to a business, but threaten it by undermining commercial confidence and customer trust. His conclusion is noteworthy; in the event of cybercrimes, victims will look for someone to be held responsible, and it will not be the hackers but the brands that the victims trusted to protect them.

Security is a never-ending challenge. As the cybercriminals evolve, so must the defenders. It's the defenders and their organizations that need to stay a step ahead of cybercriminals or else they will be held responsible for security breaches. Breaches leading to critical situations such as disclosure of customer information, denial of service, and threats to the continuity of business operations can have dire financial consequences. Yet the real cost to the organization will be the loss of customer trust and confidence in the organization's brand. Such a loss may be irreparable and impossible to quantify in mere monetary terms. Etched in the forefront of the mind of any SSLP must be the need to protect the brand, customers trust. Fundamentally, the recognition that the organization is obligated to protect the customers should powerfully motivate the organization in creating more secure software.

As a SSLP, you must protect the brand your customers trust.

Best Practice #2: Know Your Business and Support it with Secure Solutions

Most skilled security professionals agree that, along with a strong background in technology, a thorough understanding of the business is of paramount importance when it comes to creating secure solutions for that business. Though some purist security technologists may find it difficult to accept, it is nevertheless true that security is there for the business and not the other way around. Security exists to enable the business, not to be an impediment. The answer to the question, "Why were brakes invented?" could be answered in two ways: to prevent the vehicle from an accident, or to allow the vehicle to go faster. Security is similar; it can prevent the business from a crash, or allow the business to go faster.

For a SSLP, understanding the business can help in the identification of regulatory and compliance requirements, applicable risk, architectures to be used, technical controls to be incorporated, and the users to be trained or educated. For example, an internet banking organization will need to deal with regulatory requirements such as financial privacy, safeguards, and pretexting rules[c] as part of its compliance with the Gramm Leach Bliley Act (GLBA). The organization will have to address the risk of disclosure of personally identifiable and financial information, the need to have multi-factor authentication architecture, encryption, authentication, authorization, and auditing controls, as well as the need to educate employees on social engineering and phishing. A retail merchant may need to comply with the Payment Card Industry Data Security Standard (PCI DSS) depending on how they handle credit card transactions.

As a SSLP, you must know your business and support it with secure solutions.

Best Practice #3: Understand the Technology of the Software

Not only is it critical to know the business, but one must have a strong background in technology to be effective in building or buying secure software. A lack of understanding of the technology used to build or buy software can lead to insecure implementations of the software.

When it comes to building the software in-house, a thorough understanding of the existing infrastructural components, such as network segregation, hardened hosts, and public key infrastructure, is necessary to ensure that the deployment of the software will, first, be operationally functional and, second, not weaken the security of the existing environment. In other words, understanding the interplay of your current technological components with the software you build and/or deploy will help determine the impact on overall security. Further, understanding the technology used in building software can help towards making decisions that favor security. As an example, knowing that managed code (.Net and Java) have less likelihood of memory corruption and thus are less susceptible to overflow attacks than unmanaged code (C/C++), would help in choosing newer-generation managed code as part of the coding standard to develop software.

With software procurement, it is vital to recognize that vendor claims regarding the 'security' features need to be scrutinized and verified for implementation feasibility within your organization. The mere presence of security features in a product does not necessarily mean that the product is secure. The appropriate and correct implementation of the security features is what helps make a product secure.

As a SSLP, it's critical to understand the technology of the software.

Best Practice #4: Ensure Compliance to Governance, Regulations, and Privacy

In this day and age, an industry that is not regulated is more the exception than the norm as opposed to just a few years ago when the industry that was regulated was the exception. The increase in regulatory and privacy requirements imposes a serious burden on organizations. Governance, Risk, and Compliance (GRC) is not just an industry buzz phrase, but a reality and a means toward meeting regulatory and privacy requirements. As a SSLP, one must understand the internal and external policies that govern the business, its mapping to necessary security controls, the residual risk from post implementation of security controls in the software, and the evergreen aspects of compliance to regulations and privacy requirements.

As a SSLP, you must ensure governance, risk, and compliance to regulations and privacy.

"Compliance is often thought of as a finish line for an organization's security. That's the wrong mindset. Validation to compliance is merely a snapshot in time illustrating the current state of security against set criteria. Compliance should be considered as organic as an organization's business model. Even more so as the threat landscape continually evolves alongside advancements in technology."

Troy Leach, CISSP, CISA
Technical Director,
PCI Security Standards Council

Best Practice #5: Know the Basic Tenets of Software Security

When it comes to secure software, there are some tenets with which the SSLP must be familiar. These basic tenets are: protection from disclosure (confidentiality); protection from alteration (integrity); protection from destruction (availability); who is making the request (authentication); what rights and privileges does the requestor have (authorization); the ability to build historical evidence (auditing); and the management of configuration, sessions, and exceptions. Knowledge of these basic tenets, and how they can be implemented in software, is of vital importance for the SSLP.

Some mechanisms that can be implemented to support these tenets are described below. Encryption, for example, can serve as a confidentiality control, while hashing can serve as both a confidentiality control and integrity control. Encryption uses algorithms to convert humanly-readable information (plaintext) into non-readable cryptic (ciphertext) form. Decryption is the process of converting the ciphertext back into plaintext. Encryption and decryption require a key that is held secretly to perform their operations.

Encryption algorithms can be either symmetric or asymmetric. Symmetric algorithms use the same key to encrypt and decrypt and while this may be fast, the number of keys required to manage communication between parties can become cumbersome very quickly and make key management a challenge. Asymmetric algorithms use different keys for encryption and decryption, also known as a public-private key pair. Certificates are used to share the public key and the existence of a public key infrastructure is necessary. This is a lot slower than symmetric algorithms but key distribution and management is simplified.

Hashing on the other hand uses mathematical functions that are one-way. The difference between encryption and hashing is that with encryption, the original value can be re-factored; with hashing, this is not possible. Any value passed in is rehashed with the same function and then compared for validity. Hashing is more an integrity measure to detect tampering of data or files, but can be used for protecting sensitive information like passwords when stored and used for authentication.

Proper load-balancing and load-monitoring functionality in your software can protect against destruction or denial of service, and ensure availability. Password, token, or biometric means to identify and validate one's credentials are examples of authentication mechanisms, while role-based access control that the software checks is an example of authorization control. Secure software must also log and record all administrative, critical, and key business transactions so that a historical audit trail can be built when necessary. Configuration information must be treated as sensitive information and protected. Sessions should be unique to prevent any replay or hijacking attacks, and generic and laconic exception messages must be explicitly specified to prevent disclosure of too much information.

As a SSLP, it's critical to know the basic tenets of software security.

Best Practice #6: Ensure the Protection of Sensitive Information

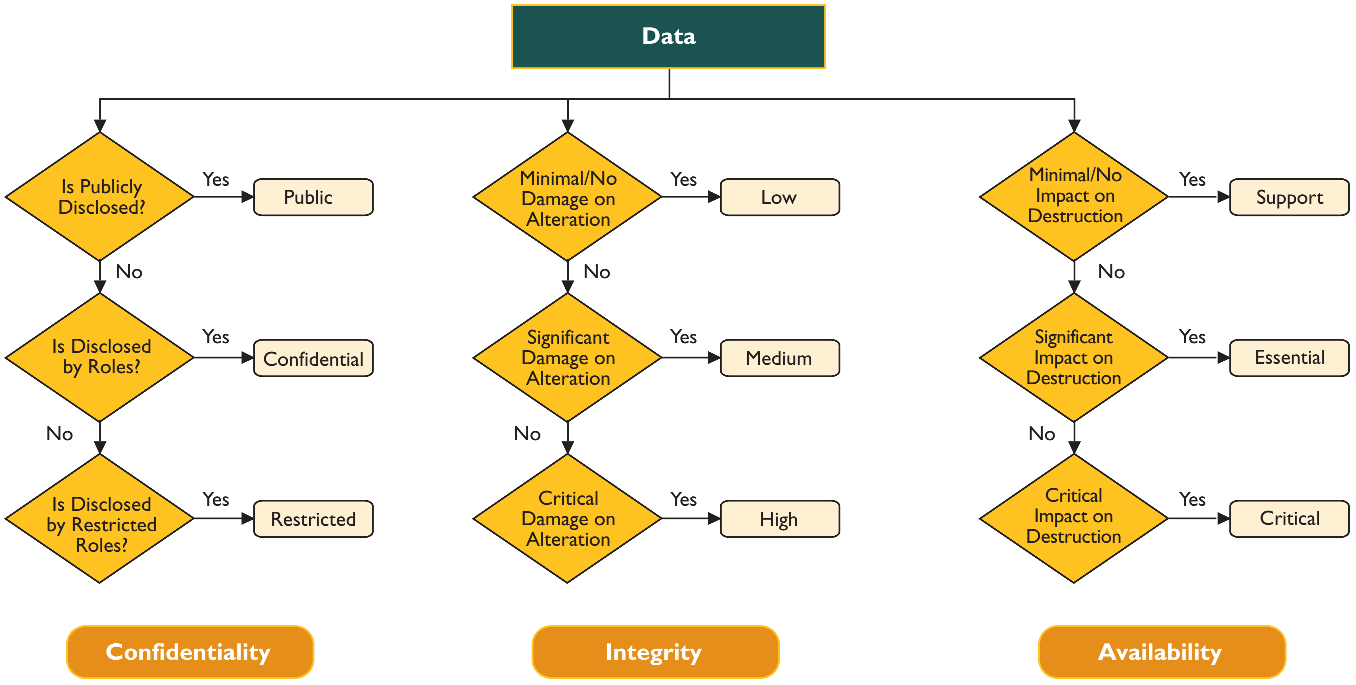
In addition to ensuring that the brand your customers trust is protected, it is essential that any sensitive information be protected as well. Sensitive information refers to any information upon which the organization places a measurable value. By implication, this is information that is not in the public domain and would result in loss, damage, or even business collapse should the information be lost, stolen, corrupted, or in any way compromised. Sensitive information may be personal, health, financial, or any other information that can affect the competitive edge of your organization.

While it is easy to identify the sensitivity of certain data elements such as, health records or credit card information, other elements may not be that evident. Determination of what is sensitive and what is not can be accomplished by undertaking a data classification exercise, with the business stakeholders involved in this process. Software that either transports, processes, or stores sensitive information must build in necessary security controls to protect this information.

A SSLP must be familiar with data classification and protection mechanisms against disclosure. Data classification is the conscious decision to assign a level of sensitivity to data as it is being created, amended, stored, transmitted, or enhanced. The classification of the data should then determine the extent to which the data needs to be controlled/secured. Figure 2 depicts an example of a flowchart that can be used to classify information.

As a SSLP, you have to ensure the protection of sensitive information.

Figure 2. Example of a data classification flowchart


Data
Is Publicly Disclosed?
Yes
Public
No
Is Disclosed by Roles?
Yes
Confidential
No
Is Disclosed by Restricted Roles?
Yes
Restricted
Confidentiality
Minimal/No Damage on Alteration
Yes
Low
No
Significant Damage on Alteration
Yes
Medium
No
Critical Damage on Alteration
Yes
High
Integrity
Minimal/No Impact on Destruction
Yes
Support
No
Significant Impact on Destruction
Yes
Essential
No
Critical Impact on Destruction
Yes
Critical
Availability

Best Practice #7: Design Software with Secure Features

The MSDN article on "Lessons Learned from Five Years of Building More Secure Software,"[d] under the heading "It's not just the code," highlights that many software security vulnerabilities are not coding issues at all but design issues. When one is exclusively focused on finding security issues in code, that person runs the risk of missing out on entire classes of vulnerabilities. Security issues in design and semantic flaws (ones that are not syntactic or code related), such as business logic flaws, cannot be detected in code and need to be inspected by performing threat models and abuse cases modeling during the design stage of the SDLC.

Threat modeling is an iterative-structured technique used to identify the threats to the software being built. It starts by identifying the security objectives of the software and profiles it. It breaks the software into physical and logical constructs generating the software context that includes data flow diagrams, and end-to-end deployment scenarios, identifying entry and exit points, protocols, components, identities, and services.

Attack surface analysis is a subset of threat modeling and can be performed when generating the software context in which sections of the software exposed to un-trusted users is analyzed for security issues. Once the software context is generated, pertinent threats and vulnerabilities can be identified.

Threat Modeling is performed during the design stage so that necessary security controls (safeguards) can be developed during the development phase of the software.

In addition to understanding how to threat model software, a SSLP must be aware of how to implement secure design principles. The well-acclaimed paper "The Protection of Information in Computer Systems"[e] by Saltzer and Schroeder lists some very sound design principles, as shown in Table 1.

As a SSLP, you must design software with secure features.

Best Practice #8: Develop Software with Secure Features

Designing for security in software is futile unless you plan to act on the design and incorporate necessary secure controls during the development stage of your software development lifecycle. It is imperative that secure features are not ignored when design artifacts are converted into syntax constructs that a compiler or interpreter can understand. Writing secure code is no different than writing code that is usable, reliable, or scalable.

"Security is just another attribute of software like usability, performance, reliability, scalability. The idea of incorporating security into the SDLC begins with evaluating the relative importance of this attribute and then going on to incorporating controls in line with that."

Tallah Mir
Sr. Program Manager
Microsoft

Table 1. Adapted from the Saltzer & Schroeder Protection of Information in Computer Systems

Design Principle	What is it?	Example
Economy of mechanism	Keeping the design simple and less complex	Modular code, Shared objects, and Centralized services
Fail-safe defaults	Access denied by default and granted explicitly	Denied transaction
Complete mediation	Checking permission each time subject requests access to objects	Credentials not cached
Open design	Design is not a secret, implementation of safeguard is	Cryptographic algorithms
Separation of privilege	More than one condition is required to complete a task	Split keys, Compartmentalized functions
Least privilege	Rights are minimum and users granted access explicitly	Non-administrative accounts, Need to know
Least common mechanisms	Common mechanisms to more than one user/ process/role is not shared	Role based dynamic libraries and functions
Psychological acceptability	Security protection mechanism unbeknownst to the end user for ease of use and acceptance	Help dialogs, Visually appealing icons

Controls, which address the basic tenets of software security, must be validated through security code reviews and security testing. It is recommended that security code review be performed while the code is reviewed for functionality, and before the software is released for testing. Security code reviews can be manual or automated. Code review tools are not a panacea and can assist only to a certain degree to identify sections of code that need attention. To keep the false positive and false negative rate to a minimum, tools often miss certain vulnerabilities. Therefore, automated code review should never be undertaken in lieu of human (manual) reviews. Definition of the scope of what is being reviewed, the extent of the review, coding standards, secure coding requirements, code review process with roles and responsibilities, and enforcement mechanisms, must be pre-defined for a security code review to be effective.

Security testing should complement existing functionality testing. At a bare minimum, tests for common software vulnerabilities, such as overflow and injection flaws, and testing the behavior of software to unexpected and random input formats (fuzz testing) should be conducted in testing environments that emulate the configuration of the production environment. Other tests for stress and performance need to be conducted as well, as they directly relate to the "availability" tenet of security. A SSLP should not only ensure that the code written is secure, but also know how to write secure code, conduct, perform, and orchestrate code reviews and security testing.

As a SSLP, you must develop software with secure features.

Best Practice #9: Deploy Software with Secure Features

Most software development teams would agree that, often, software that works without any issues in development and test environments will start experiencing hiccups when deployed/released into a more hardened production environment. Post mortem analyses in a majority of these cases reveal that the development and test environments do not properly simulate the production environment. Fundamentally, this is a configuration management issue. Changes made to the production environment should be retrofitted to the development and test environments through proper change management processes.

Another related issue is release management of software which should include proper source code control and versioning. A phenomenon that one might refer to as "*regenerative bugs*" is often observed when it comes to improper release-management processes. Regenerative bugs are fixed software defects that reappear in subsequent releases of the software. This happens when the software coding defect (bug) is detected in the testing environment (such as user-acceptance testing) and the fix is made in that test environment and promoted to production, without retrofitting it into the development environment. The latest version in the development environment does not have the fix and the issue reappears in subsequent versions of the software.

To deploy secure software, it is also recommended that software undergo vulnerability assessment and penetration testing in a pre-production environment and, if need be, in the production environment with tight control. This will help make evident the potential issues that may be uncovered by an attacker, and gives the software development team insight into the weak areas of the software.

Secure deployment ensures that the software is functionally operational and secure at the same time. It means that software is deployed with defense-in-depth, and attack surface area is not increased by improper release, change, or configuration management. It also means that assessment from an attacker's point of view is conducted prior to or immediately upon deployment. Secure design and development of software must be augmented with secure deployment.

As a SSLP, you must deploy software with secure features.

Best Practice #10: Educate Yourself and Others on How to Build Secure Software

The need to design, develop, and deploy more secure software is evident from the security incidents prevalent in the industry, and the plethora of regulations and privacy requirements one needs to comply with. The *modus operandi* of software today is the infamous release-and-patch cycle.

To combat this vicious cycle of release-and-patch, there is a need for a change – to create a culture that factors in software security from the very beginning by default. Creating a security culture can be accomplished through education. The National Institute of Standards and Technology (NIST) states that education should cause a change in attitudes, which in turn will change the organizational culture. In essence, this cultural change is the realization that IT security is critical because a security failure has potentially adverse consequences for everyone and, therefore, IT security is everyone's job.[f] Even the most expensive security measures can be thwarted by people, and educating people about software security is of paramount importance.

Not only must one be educated, but they must also be willing to share their knowledge. As Charles Dickens once wrote, "Change begets change." When one who is educated in turn educates others, there will be a compound effect towards creating the much-needed security culture.

As a SSLP, it's important to educate yourself and others on how to build secure software.

Conclusion

The McKinsey report, "The War for Talent,"[g] released in 1998 predicted that the most important corporate resource over the next 20 years would be talent. It's been a decade since the report was published, and when it comes to software security talent, this prediction could not have been any more accurate. Advancement in security technologies and improvements in processes, such as the secure development lifecycle and trustworthy computing, has accelerated. People without proper knowledge of software security can circumvent even the most carefully thought-out security implementations.

The importance of educating people and creating a culture that views software security as second nature is crucial. The newest certification from (ISC)²®, the Certified Secure Software Lifecycle Professional (CSSLP℠), is a step in that direction. Covering areas that ensure security is considered throughout the entire software lifecycle, the CSSLP is created around the specific need for building security in the software lifecycle.

Software development involves various stakeholders. Those tasked to build software securely must follow certain directives. These "Ten Best Practices for a Secure Software Lifecycle Professional" when followed will ensure that the SSLP build secure, hack-resilient, and compliant software.

People without proper knowledge of software security can circumvent even the most carefully thought-out security implementations.

About (ISC)²®

The International Information Systems Security Certification Consortium, Inc. [(ISC)²®] is the globally recognized Gold Standard for certifying information security professionals. Founded in 1989, (ISC)² has now certified over 60,000 information security professionals in more than 130 countries. Based in Palm Harbor, Florida, USA, with offices in Washington, D.C., London, Hong Kong and Tokyo, (ISC)² issues the Certified Information Systems Security Professional (CISSP®) and related concentrations, Certified Secure Software Lifecycle Professional (CSSLP℠), Certification and Accreditation Professional (CAP®), and Systems Security Certified Practitioner (SSCP®) credentials to those meeting necessary competency requirements. (ISC)² CISSP and related concentrations, CAP, and the SSCP certifications are among the first information technology credentials to meet the stringent requirements of ANSI/ISO/IEC Standard 17024, a global benchmark for assessing and certifying personnel. (ISC)² also offers a continuing professional education program, a portfolio of education products and services based upon (ISC)²'s CBK®, a compendium of information security topics, and is responsible for the (ISC)² Global Information Security Workforce Study. More information is available at **www.isc2.org.**

About the Author

Mano Paul, CSSLP, CISSP, AMBCI, MCAD, MCSD, Network+, ECSA is CEO and President of Express Certifications and SecuRisk Solutions, companies specializing in professional training, certification, security products and security consulting. His security experience includes designing and developing software security programs from Compliance-to-Coding, application security risk management, security strategy and management, and conducting security awareness sessions, training, and other educational activities. He is currently authoring the Official (ISC)² Guide to the CSSLP, is a contributing author for the Information Security Management Handbook, writes periodically for Certification, Software Development and Security magazines and has contributed to several security topics for the Microsoft Solutions Developer Network. He has been featured in various domestic and international security conferences and is an invited speaker and panelist in the CSI (Computer Security Institute), Catalyst (Burton Group), TRISC (Texas Regional Infrastructure Security Conference), SC World Congress, and the OWASP (Open Web Application Security Project) application security conferences. He can be reached at **mano.paul@expresscertifications.com** or **mano.paul@securisksolutions.com.**

a Frost & Sullivan (ISC)² Software Assurance Credential (SwAC) Study.

b *Harvard Business Review* - Breakthrough Ideas for 2008.
http://thelist.hbr.org

c The Gramm-Leach Bliley Act.
http://www.ftc.gov/privacy/privacyinitiatives/glbact.html

d *Lessons Learned From Five Years of Building More Secure Software.*
http://msdn.microsoft.com/en-us/magazine/cc16330.aspx

e Saltzer, J.H. and Schroeder, M.D., *The Protection of Information in Computer Systems.*
http://web.mit.edu/Saltzer/www/publications/protection/

f *NIST Special Publication 800-16; Information Technology Security Training Requirements: A Role- and Performance-Based Model.*
http://csrc.nist.gov/publications/nistpubs/800-16/800-16.pdf

g *The War for Talent*, McKinsey
http://www.mckinseyquarterly.com/The_war_for_talent_305

(ISC)^2®