Secure System Development Life Cycle (SDLC)

Building Security Into Your System--
Not Bolting It On After the Damage is Done

By Patrick McBride
& Edward P. Moser

Copyright Notice

METASeS
8601 Dunwoody Place, Suite 700, Atlanta GA 30350
678-250-1250 fax: 678-250-1251
www.metases.com

Printed in the United States of America.

Warning and Disclaimer

Table of Contents

Introduction

This section outlines the key terms, premise, purpose, audience, and background of the report.

Audience

The audience for this report is primarily members of application and infrastructure development teams.

The security team in an organization will often explain, to the development, infrastructure, and business teams, the importance of having a plan to build security into the life cycle process. We've often found that the security personnel then "fall victim to their own marketing success". The teams in question, duly lectured, and enlightened, will often then request guidance on how to accomplish the task. These kind of requests are growing more frequent as computer security becomes a more pressing need.

In addition to enhancing communication between the security and application development teams, a security framework also can be used to better define requirements for outside consultants responsible for system development initiatives.

This report assumes a certain level of understanding of System Development Life Cycle (SDLC) processes, but not necessarily a comprehension of security issues. We define any security-related matters that arise in the report.

Key Terms

Important terms contained in this report are defined below.

Asset Classification – Classifying and labeling of information so that users understand its value to the organization, and what they need to do to protect it.

Assets – The type and value of data or information to be protected. Assets refer to the data contained in or traversing computer networks.

Asset Value – The value of the different assets that you are trying to protect. The worth of the data that intruders target.

Controls – Technical and non-technical measures put in place to eliminate or mitigate risk.

Denial of Service (DoS) – The inability of a Web site to function for an extended period.

Risk – The likelihood of loss, damage, or injury. Risk is present if a threat can exploit an actual vulnerability to adversely impact a valued asset.

Risk Management – Identifying, assessing, and appropriately mitigating vulnerabilities and threats that can adversely impact the organization's assets.

SDLC – The integrated, iterative process of analyzing, designing, developing, deploying, and enhancing applications or infrastructure, including both third-party and in-house applications.

System – In the context of this report, refers to both applications and infrastructure (hardware, operating systems, software, etc).

Threat – The actual people or organizations that could exploit a vulnerability (or hole) in your organization and put your information assets at risk.

Threat Analysis – Evaluation of would-be attackers and of the damage that they are likely to purposefully or accidentally perpetrate.

Vulnerability Assessment – Activity in which an organization identifies and prioritizes technical, organizational, procedural, administrative, or physical security weaknesses. A vulnerability assessment should yield a traceable, prioritized "road map" for mitigating the assessed vulnerabilities.

Premise

The growth of the Internet and e-Commerce is taking place at an exponential rate. From 1998 to 2003, according to industry analysts, business-to-business sales will rise from $131 billion to $1.5 *trillion*. The security risks to business are comparably dramatic. Companies are responding to the growing threat by purchasing more security applications.

Before the emergence of the Internet, most organizations, in Information Security terms, resembled castles with very big moats. Except for occasional forays outside, the employee "foot soldiers" stayed inside the walls. The only way a hostile force could enter the premises was by storming the walls, that is, by physically breaking in. Today, organizations are akin to armies in the field, ever on the march through the outside world. In-house employees are in constant electronic interaction with customers and employees off-site in distant lands, and exposed to ambush from shadowy intruders slipping past electronic sentries and barriers.

The transition from the "old economy" to the "new economy" reflects these new security concerns. In the past, airline company employees, for example, would make all the passenger reservations themselves through a back-end computer system. Today, the passengers themselves make the reservations, through a Web front-end that is integrated through the back-end system. Similarly, in the manufacturing environment, partners are linking their systems via sophisticated supply chain networks. In essence, customers and partners – and potentially malicious intruders – have been granted access to the electronic fortress.

Despite a greatly altered security landscape, the security aspects of computer applications development are too often ignored, given low priority, mishandled, or simply misunderstood. Many application development and business teams do not understand or know how to handle the new risks. Quite a few development teams and business teams – in human resources, manufacturing, finance, etc. – are not properly trained in the security aspects of application development. Some are aware in a general way or at a high level of increased security risks, but not at the detailed or application levels that involve knowledge of security procedures and techniques.

Further, many organizations often treat security as an afterthought. This approach leads to insecure applications, and adds considerable expense when security fixes are retrofitted onto existing systems.

To date, most security activity has focused on securing infrastructure such as firewalls or network access controls. Although the base infrastructure is very important, it does not cover every security concern. Developers must realize that on-line attackers can not only penetrate networks, but can misuse applications. Thus, organizations need to put just as much effort into securing applications, and must design security into applications. (For detailed information on application security, see the METASeS report, *Building Secure e-Commerce Applications*.)

It is vital to put non-technical controls on an equal footing with technical controls. ("Controls", both technical and non-technical, refer to measures put in place to eliminate or mitigate risk.) Non-technical controls such as training and education are just as important as technical controls. Yet all too often the attitude is, "All we have to do is put in a firewall, and we'll be secure." .

Security teams have to work more closely with the application, infrastructure, and business teams to train them on defending against "vulnerabilities", that is, the "holes" in your organization – the potential weaknesses in information systems and procedures that intruders and threats can exploit.

An initial step to take is to make the application teams aware that security is an issue. They must learn to make this an overt design point in new systems. A follow-on step that many organizations should take is to provide assistance, such as training, architecture assistance, and review teams, in application-level security.

At the same time, it is important for application developers to realize that not every risk should be handled the same way. Not everything in your organization can be or should be completely protected against attack. You will want to fashion an "appropriate" level of security controls based on the value of an asset and the risk posed to it.

In short, security needs to incorporate both technical and non-technical controls. Most importantly, it must be integrated into the applications and infrastructure instead of bolted on afterwards. It should be addressed in each phase of the life cycle, from requirements planning to maintenance and operations. If you do not address a concern in its respective phase, it will only pose more of a risk and become more expensive and more difficult to fix later on.

Purpose

This goal of this research paper is to help ensure that systems have security that is both *appropriate* and *adequate*.

"Appropriate security" means security that is geared to an application's level of risk. As George Orwell wrote in his classic satire *Animal Farm*, "All animals are created equal, but some animals are more equal than others." The same applies to computer systems and the data/information assets they house. Some systems are much more valuable, and thus require more resources devoted to their protection. A corporation might be little concerned about protecting a computer file with records on meetings from two years prior. But a blood bank would be very concerned about protecting a database containing the medical history of its donors. Similarly, a police department would want to achieve a high level of security for files on Mob investigations, or for internal investigations of its own officers. Systems containing Defense Department information have very specific requirements based on the information classification. The amount of security built into a system is directly related to the importance of the data to the organization. In other words, the criticality of the data, and determines the organization's risk tolerance toward it.

A safety belt installed on a poorly designed car will not make the car safe. Similarly, a firewall will not sufficiently protect a system that was not designed with appropriate security in mind. The point

is that security must be built into new systems and applications, rather than "bolted on" afterwards at considerable trouble and expense. A comprehensive, systematic approach to implementing security from the very start of applications development is essential. Organizations need a blueprint for building security into applications development, that is, a schema they can incorporate into every phase of the SDLC.

To accomplish this aim, we outline the tasks that organizations need to perform in each phase of the *System Development Life Cycle* (SDLC). The SDLC refers to the integrated, iterative process of analyzing, designing, developing, deploying, and enhancing applications or infrastructure, including those purchased or generated in-house.

For each phase of the SDLC, we spell out the issues to raise, the tasks to accomplish, and the output to generate. Our focus is on applications developed in-house. In later versions of this document, we will discuss packaged applications – ones purchased commercially off-the-shelf (COTS), for example, SAP and PeopleSoft.

While this paper often discusses Web-based applications, it is also appropriate for more traditional (legacy) applications. It especially emphasizes the requirements analysis and architecture/design phases of the SDLC – areas that organizations most typically overlook in developing systems.

You can employ this document in two broad ways:

1. Use it as a companion document, on a project-by-project basis, to superimpose a security "overlay" on your SDLC processes.

2. Use it as a master template to incorporate security-related functions into the organization's existing SDLC.

Organization

This paper is composed of:

1. Narrative text
2. Appendix A – SDLC Security Overlay Checklist
3. Appendix B – SDLC Overlay
4. Appendix C – Security-related Processes and Procedures

The body of the document discusses, for each phase of the SDLC, key aspects of security that you need to incorporate into the life cycle. These cover tasks that are relatively more important or that need amplification. They are the minimum actions you need to take: we do not attempt to cover every security task.

Appendix A is a key part of this document. It supplies a comprehensive checklist of security-related tasks that should be performed during each SDLC phase. It will help you ensure that appropriate security is designed or built into specific systems.

We've organized the body of the document and Appendix A sequentially by phases – requirements analysis, design, development, testing, etc. Each phase is broken down into security-related tasks you should perform for it. Appendix A also explains why certain actions should be performed, who should perform them, and what the output of the actions should be.

The report contains two other appendices. Appendix B contains an overlay for Microsoft's commercial SDLC. Future versions of this report will provide overlays of other SDLCs. This appendix basically provides a mapping between our "typical" SDLC and the Microsoft SDLC. Appendix C contains a list of common security-related processes and procedures.

Background

In today's environment, software systems are much more vulnerable to security risks. There is an ongoing explosion of new tools, applications, and technologies. At the same time, vendors usually do not adequately test new applications for security defects.

Present-day systems have more "moving parts", such as Graphical User Interfaces (GUIs), middleware, network connections, etc., that make them more susceptible to intrusion by malicious users. An analogy to make is with the electric motor and the internal combustion engine. An electric motor has few moving parts, can run for long periods without mechanical problems, and needs little maintenance. The internal combustion engine, on the other hand, has many moving parts, breaks down frequently, and requires a lot of maintenance at the repair shop. In a related way, older computer applications have fewer parts, and tend to be easier to secure. Newer, Web-based applications have many parts, from many vendors, and are therefore more susceptible to security vulnerabilities.

Therefore, ensuring security nowadays is a complicated process. Figure 1 illustrates this.

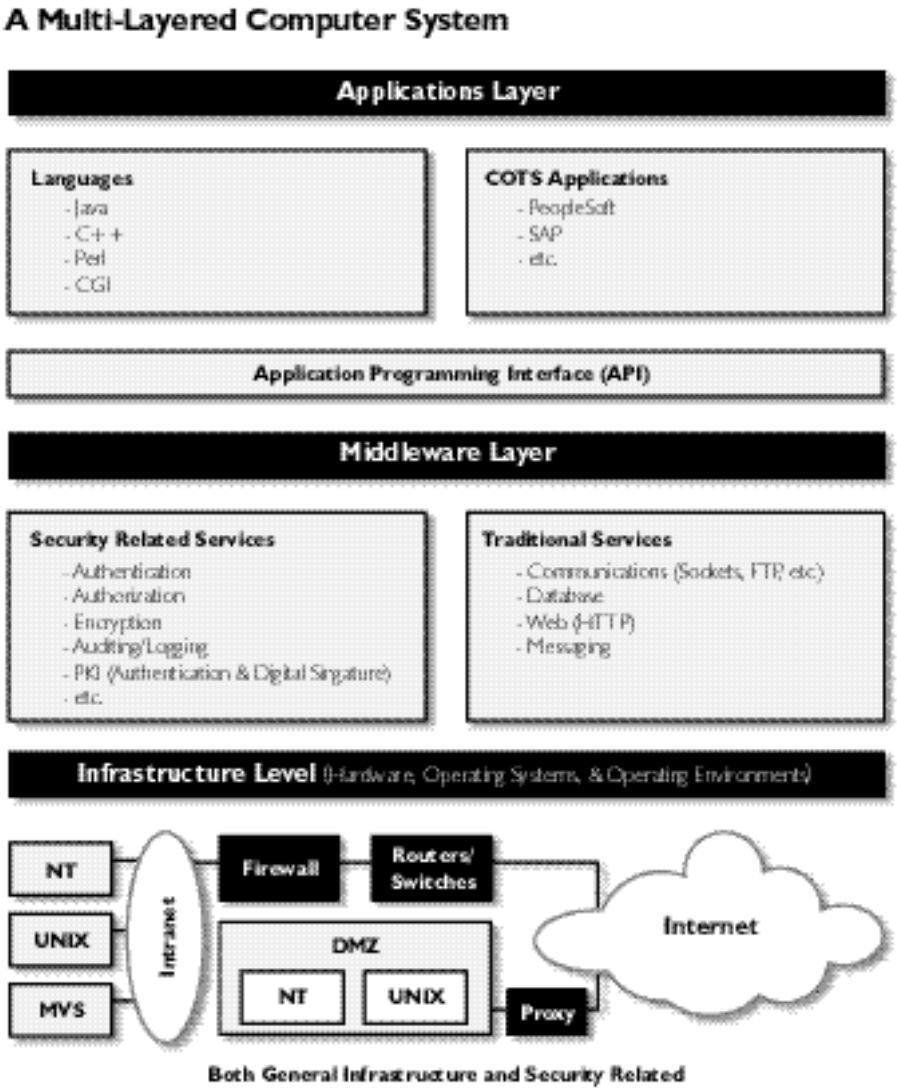


Figure 1

It depicts the layers of a typical system: the applications layer, application programming interfaces (APIs), middleware layer, and infrastructure layer. (Note that the figure is illustrative; there is no such thing as a perfect application model.) The problem from a security perspective is that, for any of the layers, multiple components for multiple internal and external sources need to work together securely.

Each section of the figure is explained below:

- **Infrastructure –** Includes hardware (routers, servers, etc.) and operating systems (for example, NT, UNIX, MVS). There are two types of infrastructure: general infrastructure and security-related infrastructure. Some components, such as firewalls, relate specifically to security.

- **Application Programming Interfaces (APIs) –** Applications, whether programming languages or commercial-off-the-shelf (COTS) systems, use a set of APIs to call and receive various middleware and infrastructure services.

- **Middleware –** Middleware can be divided into traditional and security-related services. Traditional services include communications services (sockets, FTP, etc.), Web servers (HTTP), and messaging services, for example, queuing, and store and forward. Certain items are security-related, for example encryption and authorization.

- **Applications –** Applications include languages such as Java, C++, Perl, COBOL, and CGI. They also include COTS applications such as PeopleSoft and SAP.

In the past, most security attention focused on network infrastructure layers as opposed to application layers. Application layers were much less of a concern because they were typically not exposed to external attack. But today, due to the growing use of networks and the growing dominance of the Internet, intruders can penetrate an organization's infrastructure through the application layer as well as the network infrastructure layer. Moreover, after an attack, they can misuse or destroy data and make applications perform unintended functions. Security, therefore, needs to address the various layers of the system as a whole.

SDLC Phases

There are various kinds of SDLCs. This section discusses the following two major types:

1. Waterfall
2. Rapid Application Development (RAD)

Refer to Figures 2 and 3.

In an ideal world, development teams generate a complete and perfect set of requirements to smoothly drive the design, development, and other phases of the life cycle. This sort of process may be represented by the waterfall application development process. It is characterized by an orderly, separate, and sequential flow of tasks performed in discrete phases, from initial concept, to design, development, testing, implementation, maintenance, and revisions.

Unfortunately, this kind of problem-free, uninterrupted process rarely takes place in today's development setting. More typically, the development team has to jump back a number of times to previous phases in order to fix incorrect, overlooked, or incomplete parts of an application.

A related concern arises from the often errant presumption that the business team sponsoring an application can clearly describe what it wants and needs. It frequently has a good idea of what the application should accomplish, but is usually unable to outline the application in adequate detail.

One way to grapple with these issues is to use an iterative or circular development technique, the so-called rapid application development (RAD). RAD uses iterative, cross-functional development techniques, and program tools such as Visual Basic, to develop prototypes and quickly design new systems. RAD evolved out of artificial intelligence (AI) systems in which it was difficult for subject matter experts to explain their complex knowledge in such a way that it could be readily coded into applications. This challenge led to the use of prototypes and iterative development. Later, RAD methodologies were adapted to the development of traditional, non-AI systems.

Companies can adopt various commercial incarnations of SDLCs, for example, Microsoft's Microsoft Solutions Framework (MSF) (see Appendix B).

Some modeling techniques, for example, Unified Modeling Language for object-oriented development, also contain life cycle elements. While they are not SDLCs themselves, they often provide development project guidance in addition to modeling techniques.

While each system development process differs within phases, it generally adheres to the standard life cycle phases. Some may follow the waterfall model, others the RAD model, and still others a hybrid of the two. Appendix B overlays our general SDLC onto Microsoft's SDLC.

The various models use different terminologies, but follow the same basic phases, as outlined in the next section.

Note that, for various functions, many organizations purchase commercial off-the-shelf (COTS) applications instead of developing their own applications. Security-related SDLC variations for these "buy and integrate versus build" projects are covered at the end of this document.

Waterfall Life Cycle Development

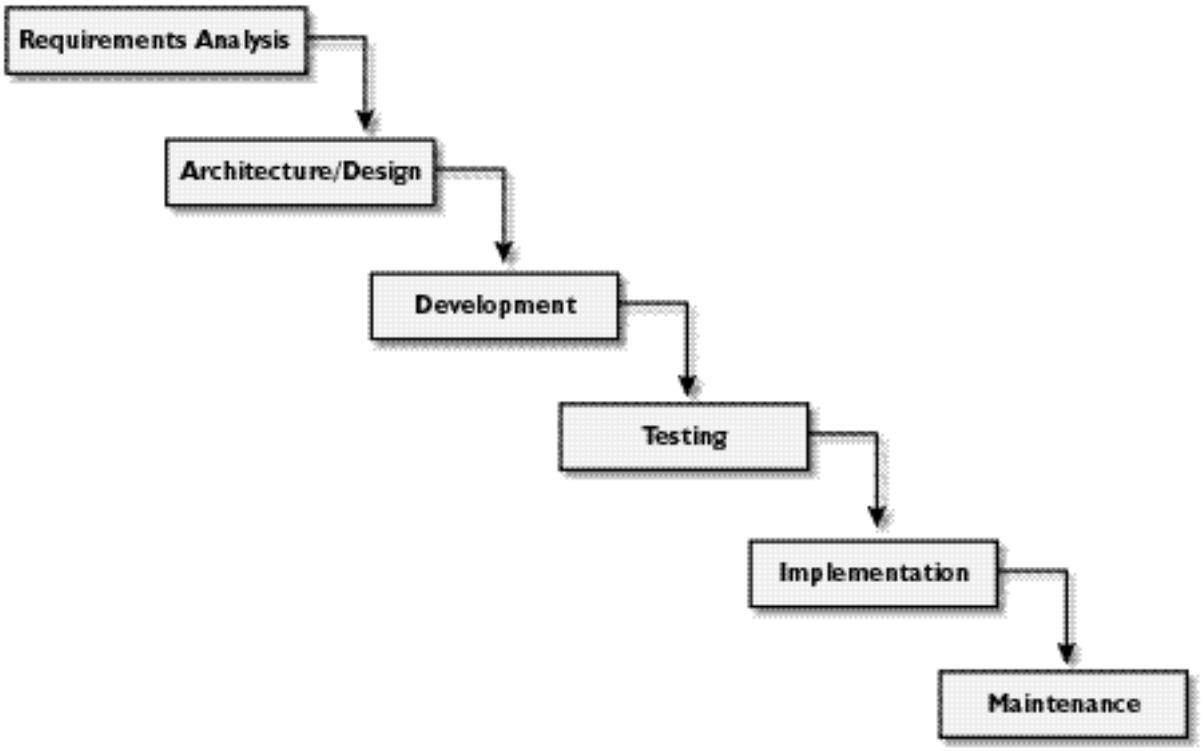


Figure 2

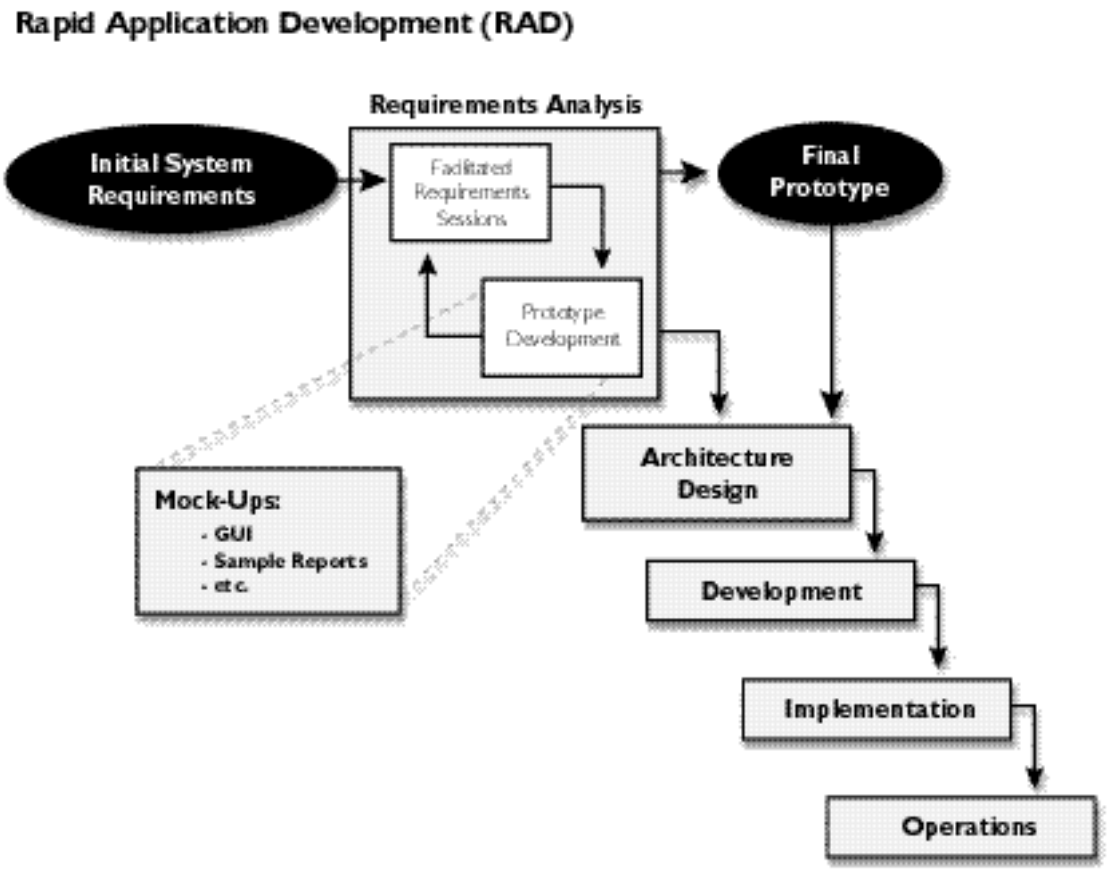


Figure 3

SDLC Phases

A typical SDLC has six phases:

1. Requirements Analysis
2. Architecture and Design
3. Development
4. Testing
5. Deployment/Implementation
6. Operations/Maintenance

From a security perspective, each phase has its own set of tasks that need to be accomplished to infuse the appropriate level of security into the final system. Each phase and its set of tasks is discussed in turn. For each, we define the phase, explain in general terms what we are trying to accomplish from a security perspective, and discuss key security-related tasks.

We won't get into the many non-security-related tasks that application developers face; we focus on security alone. We discuss the relatively more important tasks, and ones that need further elaboration. Appendix A includes a more comprehensive list of security-related tasks.

Requirements Analysis

When you build a home, you first determine your requirements for the house before settling on particular features. You would decide on, for example, whether to build a landscaped house or a townhouse. Then you would choose the number of rooms, the kind of heating system, the amount of open space, etc. In a similar way, the requirements analysis phase identifies the parameters for building a system by defining the business requirements for the system, its number of users, types of data, required output, etc. It defines what the system is supposed to do by providing high-level statements of system functionality.

The security-related requirements analysis is typically performed in facilitated sessions between the application developers and the business teams.

Key security-related tasks in requirements analysis are:

1. Analyze security requirements.
2. Discuss business operations.
3. Profile users for the organization's applications/systems.
4. Develop prioritized security solution requirements.

Analyze Security Requirements

Analysis of the security requirements is a critical task, in part because application teams either ignore or get this function wrong. All too frequently, they put the "cart before the horse", and design a security solution before tackling the true business-level security needs.

You want to bring together business managers and application development and security team executives to better understand the key sensitivities and business consequences of Information Security risk, including various breach scenarios in which information is illicitly accessed or damaged. Your aim is to gather the information required to drive the next phase – architecture and design.

A security requirements analysis may be split into these subtasks:

1. Evaluate security risks and consequences.
2. Perform asset value analysis.
3. Discuss the potential threats.
4. Analyze potentially malicious or harmful activities.
5. Analyze high-level vulnerabilities.
6. Discuss the security goals.
7. Review regulatory requirements and corporate Information Security policy.
8. Review future business goals.

Each subtask is discussed in the sections that follow

For the output of the security requirements discussion, you begin to determine what security risks your business should be concerned about, that is, what you want to protect against. The subtasks are sequenced in such a way to help guide the business team into articulating their needs, thus providing the basis for a prioritized requirements document. The prioritized requirements document will logically lead you into the next phase, architecture and design.

1. Evaluate Security Risks and Consequences

The real risks to your business need to be understood, both by business personnel and by application owners.

This process is in part educational. Executive teams normally have considerable expertise with risk management, but do not necessarily possess much knowledge about Information Security. Rolling through the types of security risk and the associated business consequences can constitute a bracing primer on this subject.

As a start, you should examine the main classes of security risk, as identified in following table.

Main Classes of Security Risk

Sample Information Security Risks	Consequences
Fraud	Loss of current revenue Loss of customer trust and future revenue
Theft	Loss of revenues
Destruction	Reduction in future revenue Loss of customer trust
Breach of Privacy	Loss of future revenues Loss of earnings Legal liability and expenses
Denial of Service	Loss of current and future revenue

Table 1

The classes of security risk are briefly defined below.

Fraud – False representation of a product or service with the intent to steal or falsely solicit funds or assets.

Theft – Stealing of information. Crackers might access, for example, customer credit cards to conduct purchases under false pretenses. Industrial espionage agents may steal future product plans or other intellectual property and sell them to commercial competitors or foreign governments.

Destruction – Damage to your Web site or database. A common act of destruction is the defacement by crackers of Web sites. Perhaps the most feared risk is the destruction of your hard drives by intruders.

Breach of Privacy – Public disclosure of sensitive customer information. The effect of breach of privacy can be devastating to an online business. Imagine the effect on an online book and video store of crackers posting the buying habits of customers. Or the effect on an online drug store of crackers posting the medical test results of clients with sensitive ailments such as AIDS. The impact on your bottom line, where vendors are changed with just a click of a mouse, can be huge.

Denial of Service (DoS) – The inability of your Web site to function for an extended period. In early 2000, the launching of undetectable DoS attacks against a number of high-profile Web sites made headlines. DoS attacks can literally cripple an online business dependent on the Internet for interaction with its customer base.

Before attempting to architect and design technical and non-technical controls, it is critically important to know what needs to be protected (assets), and what you need to be protected from (threat and activities), and what the resulting business consequences would be if protective measures are not taken or prove inadequate. For commercial businesses, there are many consequences, all funneling down to either revenue impact or profit impact, or both. In the non-commercial world, the loss of reputation or trust from employees and constituents is another major consequence.

2. Perform Information Asset Value Analysis

Let's start this discussion with some definitions. Assets refer to the type and value of data or information to be protected. In this context, assets refer to the data contained in or traversing computer networks, not the computer networking equipment itself. Asset value means the value of the different assets that you are considering trying to protect. It refers to the relative worth of the data that intruders target.

In this analysis, we are not primarily concerned about the cost of the underlying hardware and software systems, but with the value and sensitivity of the data riding or residing on them.

You want to perform an analysis of the information assets associated with the system. This is a key step that is sometimes overlooked. You have to know what you're trying to protect, and what it is worth to your organization, in order to appropriately protect it. A key tenet of Information Security is that you cannot protect 100 percent of your assets with 100 percent assurance. Again, although in theory all assets are equal, some assets are more equal than others. The steps or controls taken to protect it have to be in accord with the value of the assets. The goal is to ascertain what information assets and even intangible assets (such as corporate image and investor confidence) are most valuable and worthy of protection.

Defining the important items that need protection will help weed out unnecessary security goals, tighten the requirements, and help your organization formulate its security budget.

A complementary exercise is asset classification, the classifying and labeling of information so that its users understand its value to the organization, and what they need to do to protect it. Organizations typically have a classification scheme with different levels, for example, for proprietary and confidential.

3. Discuss Potential Threats

One key input that is often overlooked, or given too little attention, is threat. Again, threats refer to actual people or organizations that could exploit a vulnerability or hole in your enterprise and place at risk your information assets.

You have to understand the potential threats in order to devise a practical security solution. The security solution you devise will in part be a function of the level and likelihood of the threat. (If you lived far out in the countryside, for example, the danger of someone breaking into your house would be minimal, and you might not bother to lock your doors. If you lived in the middle of a big city, where there were many burglars, and lots of other threats, you might bolt and padlock every door and window.) In the computer realm, crackers engaging in random pinging of your Web site – the cyber equivalent of a thief driving through a neighborhood looking for targets - would constitute a frequent yet low level of threat. In contrast, a foreign espionage agent or organized crime ring, that targeted a particular prized asset, would constitute a high-level threat. In some other situations, the threat will be small enough to warrant a minimal response, or no security controls at all.

4. Analyze the Potential for Malicious Activities or Accidental Harm

Earlier in this phase, during discussion with the business teams, you cover general Information Security risks. The goal is to bring the business team up to speed on Information Security. In this subtask, the key is to understand more specifically what potentially damaging activities need to be protected against. A key goal is prioritization of these activities. Having just discussed threat, we now turn to analyzing and prioritizing the activities where threats are likely.

These activities can be either malicious or inadvertent, and from internal or external sources. They include activities like Denial of Service, theft, destruction, and alteration of data. In addition, they include the chance of accidental destruction or alteration of data, or the repudiation by end users of transactions, among many others. From an information protection perspective, some of the security controls you architect and design will help mitigate or avoid many of these activities. The key is to draw up a comprehensive, prioritized list so that controls address the most likely threatening activities.

5. Analyze High-Level Vulnerabilities

Vulnerabilities, to restate, are holes in the system or organization that threats can exploit to impact the organization's information assets. At this juncture, it is not important to discuss each of the low-level vulnerabilities. Rather, it is important to understand the various classes of vulnerabilities that the system-level security architecture must address. Some high-level classes of vulnerabilities are:

- **Education/Awareness-Related Vulnerabilities** – Lack of security awareness from end users, developers, IT infrastructure, or operations teams, etc. causes potential holes. Examples include using easy-to-guess passwords, or posting passwords on yellow sticky notes attached to terminals. They also include susceptibility to "social engineering" attacks where intruders might pose as legitimate personnel, for example a security administrator, in order to obtain sensitive information or gain additional access.

- **Configuration Vulnerabilities** – Threats can exploit poorly configured software or hardware systems to gain direct access to or to sniff out the weak links in a system. Examples include missing or incorrectly configured security controls or the running of unnecessary or potentially vulnerable services on a susceptible network system.

- **System Vulnerabilities** – Sometimes jokingly referred to as "unplanned features", these vulnerabilities occur when either hardware or software vendors or applications development teams inadvertently, or sometimes purposefully, incorporate holes in the system. This issue was prominent during the Y2K remediation, as many organizations hired outside contractors to debug programs – and hopefully to not add "back doors".

6. Discuss Security Goals

After analyzing general risks, asset valuation, threats, specific activities and high-level vulnerabilities, you can then prioritize the security goals for the system. There is a limited set of security goals. As with analyzing potentially malicious activities, the key is to prioritize the goals so that the security solution designed in subsequent phases meets the objectives.

The security goals are:

- Confidentiality and Possession
- Integrity and Authenticity
- Availability and Utility
- Non-repudiation
- Auditing

The following text provides definitions for these key security goals. Figure 4 illustrates the security goals.

Confidentiality and Possession
Confidentiality is the goal of keeping private the organization's vital information, such as partner or customer data in custodial care, and intellectual capital, which in many cases is the core, competitive advantage of the business. Possession involves data that a user owns, but is not necessarily familiar with or aware of. It especially true in an increasingly computerized world with massive amounts of information, all of which a user cannot possibly hope to be familiar with.

The chief methods for achieving confidentiality and possession are:

- **Authentication** – Provides the means for identifying systems or people through the proper credentials. The tools that enable this method include simple ID/password combinations, software or hardware tokens, two-way authentication, biometrics, etc.

- **Authorization** – Used in parallel with authentication to protect system or network resources. The authorization process will use the information that has been captured by the authentication process, such as user ID, group, domain, etc., to allow access to resources based on permissions and rights granted to the entity. The tools that can facilitate authorization may be a collaboration of authentication, system tools, and application code.

- **Encryption** – Helps to protect data that is transmitted across a network or stored locally on a machine. It scrambles the data to render it unintelligible unless it is decrypted, or unscrambled. In support of confidentiality, encryption ensures that credentials or personal and system identification information (keys, passwords, addresses, etc.) that are used to authenticate and authorize a person or system cannot be compromised. Some tools to accomplish this include encryption algorithms such as triple DES or protocols such as IPSec.

Integrity and Authenticity
Integrity refers to the consistency of the data. The goal specifically relates to the need to ensure that an unauthorized person or system cannot inadvertently or intentionally alter data. Imagine a database of pricing information where the data stored is not inspected properly and could be altered for the financial advantage of the perpetrator, or a financial report on a Web site that is intentionally altered for stock trading advantage.

In an e-Commerce environment, integrity is a critical factor in the validity of the data being exchanged and stored. To ensure data integrity, cryptographic algorithms and data inspection techniques can be used. Tools such as cryptographic hash functions (e.g., MD5 and DH), in combination with sound programming techniques for data inspection, allow you to check data integrity.

Authenticity refers to the ability to know that the information accessed is genuine. An example of a breach of authenticity would be a perpetrator falsely posting an official-looking company press release in a public place like a news portal. Digital signatures and other hard-to-duplicate techniques, such as holograms, can help ensure the authenticity of information.

Availability and Utility
Availability refers to the accessibility and usability of information. It is essential for an organization that provides e-Commerce to incorporate availability in the initial design and deployment of the network. Imagine the customer dissatisfaction and loss of revenue if an on-line store were disrupted from servicing customer orders during the Christmas season – especially as the Internet places the customer only a few clicks away from a competitor.

Utility refers to the usefulness of information. By utilizing new cryptographic techniques, organizations can protect information from confidentiality breaches. However, if keys are lost then the information may be useless.

Methods for achieving availability include backup of data, redundancy, and fail-over methods that provide continuity of services. The tools that can help achieve availability include:

- Local or network backup systems
- Cold or hot stand-by equipment that can be swapped or automatically activated to start processing information
- Mirroring that allows data to be available in physically separate sites, thus minimizing the risk of single point of failure
- Load balancing techniques that help minimize the impact of potential system failures

Utility can be achieved by maintaining backup copies of information and the various keys used to encrypt them.

Auditibility
Auditibility refers to the necessity of keeping historical records of events and processes such as authentication, resource access, or data exchanges. These records can make it possible to detect inappropriate use of resources or signs of intrusion. They must be handled with the proper care (e.g., stored in a safe place, digitally signed, encrypted) to be admissible in court, in the event they are used to prosecute someone who stole or damaged your data.

Event logging is a method for recording historical data. Typically, computer systems are configured by default to record general events such as operational errors, access requests, failures, or successful logins. Although this provides some accounting capabilities, it may not provide the required level of detail to recreate a clear picture of security events. A system that maintains customer information may be configured to record only general events such as program errors or run-time library errors. In fact, a system with customer data may require additional security logging.

Non-Repudiation
Non-repudiation provides evidence for verifying the identity of an entity (person or system) engaged in a business transaction, such that it cannot deny being a party to the transaction. In the non-digital world, this is typically accomplished through a signature and, in certain instances, backed up by a third-party witness such as a notary public.

A typical instance of non-repudiation involves a person sending a message, and the recipient having to verify that the originator is the author of the message. Further, the originator must not be able to deny ownership of the message. For example, in the event of a dispute with a customer, an on-line brokerage firm must be able to prove in court that transactions submitted to purchase or sell stock are authentic and came from the person in question. Cryptographic methods such as generating digital signatures or digital fingerprinting help implement non-repudiation.

You must incorporate the goals of confidentiality and possession, integrity and authenticity, availability and utility, non-repudiation, and auditing, while building and maintaining any computer system, especially systems that provide e-Commerce services prone to attack. They are the fundamental business requirements of the security architecture.

Understanding these fundamental security goals, and their fundamental priority for a given system, will help the organization formulate a sound security solution. It is one of the most important inputs to the next phase – defining an appropriately secure architecture.

It is important to clarify your goals at this point, during the discussion of security goals, before you start devising methods and tools to accomplish your objectives. Avoid discussing the nitty-gritty of technical solutions during this time. All too often, the team will focus on specific security methods or tools such as PKI (public key infrastructure), before deciding what the broad goals are. This puts the technological cart before the horse – avoid that!

You must determine your high-priority goals. For example, in an online drugstore, where large amounts of pharmaceutical products are sold online, protecting customer records is paramount. In any online business, the availability and integrity of the Web site is key. It is critical to prevent crackers from taking down the site (availability), or from placing unauthorized material on it (integrity).

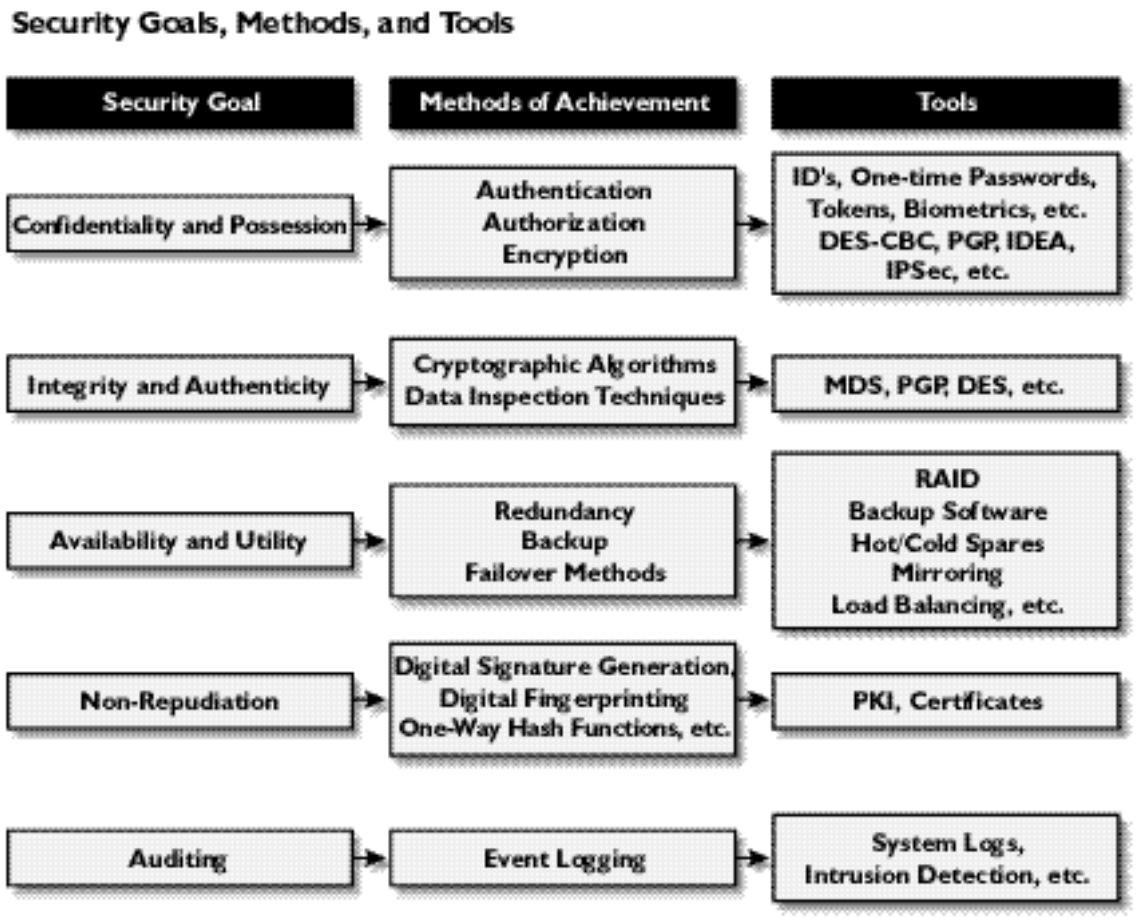


Figure 4

Executives must exhibit "due care", that is, take prudent steps to understand the security requirements and goals required to mitigate key business risks and avoid negative consequences. Corporate executives also have a fiduciary responsibility to shareholders to protect the value of the business. In short, they must "protect their assets while covering their asses." Therefore, articulating and prioritizing the security goals is a crucial and fundamental step.

7. Review Regulatory Requirements and Corporate Information Security Policy

An often-overlooked chore is to understand the regulatory requirements affecting the computer-related aspects of your business.

It is important to review previous audit findings by either internal or external auditors. You should examine which conditions to mitigate, which to leave alone, and which are likely to worsen.

For example, with an online drugstore, a key set of requirements is the Health Insurance Portability and Accountability Act (HIPAA). It sets stringent requirements on the confidentiality of health care information. Your business may be liable for expensive lawsuits if crackers get their hands on customer health care data. According to HIPAA, all health clearinghouses and healthcare providers that transmit or maintain electronic health information must conduct a risk assessment and develop a security plan to protect this information. They must document these measures, keep them current, and train their employees on appropriate security procedures. Wrongful disclosure of individually identifiable health information is punishable by $50,000, imprisonment of one year, or both.

Airline and aerospace firms must comply with strict Federal Aviation and Administration (FAA) guidelines. Publicly traded firms face tight Security and Exchange Commission (SEC) rules against premature disclosure of financial information, especially information that can have a material

impact on stock prices or public offerings. To the extent that such data could be accessed illicitly over the Web, it must be tightly safeguarded.

While reviewing the regulations affecting your business, don't forget to make sure that your team is also following your corporate Information Security policy (refer to the link at the top of this document).

8. Review Future Business Goals

Current and future business goals refer to what the business system and information owners want to achieve with the system today and down the road. This activity ensures that the team, by considering the future business requirements during the architecture and design process, does not build itself into the proverbial box. An example of such an error would be a system model that originally accounts only for internal users (employees and partners), even though the business goals envision opening the system to external customers. An unfortunate result may ensue: authentication techniques such as tokens (e.g., SafeWord Soft Token, Security Dynamics, etc.) that might work for internal users could fail for an external audience consisting, potentially, of everyone on the Internet.

Discuss Business Operations

The next major task in supporting the security of the enterprise is to understand the business operations. Knowing the processes and procedures that constitute business operations will help you understand the vulnerabilities facing them. Additionally, you can begin to realize how other, related security operations fit into the equation. For example, security administration, which involves among other things adding and maintaining user privileges, needs to be integrated with various business processes, such as adding and servicing customers and partners.

You will also consider what other non-technical vulnerabilities may exist, and how you might mitigate them (for example, through audit reviews, dual entry, or separation of function). To accomplish this task, security personnel and members of the application development team should meet with mid-level business managers. One output of this task should be a list of business processes with potential vulnerabilities that require controls, for example, credit card transactions. A second output should be processes that need to incorporate security processes and procedures, for example, the adding and maintaining of customers. Another example is the IT change management process for placing new systems into production environments. Note that both of these lists may overlap.

Profile Users for the Organization's Applications/Systems

An important task in defining requirements for any system is determining who its users are. Specifying the users helps determine key project scope issues and design considerations, including those of a security-related nature. For example, if only internal users are to access a system, you'll be far less concerned about external access control issues. An online drugstore might have a limited set of users, for example, a certain number of regional centers, or bulk customers such as hospitals or nursing homes. A public key infrastructure (PKI) scheme might be a good solution for such a limited set of customers. On the other hand, a Web site that sells pharmaceutical products to users throughout the world demands a different kind of authentication scheme because of its larger scope, far greater volume of users, and changing set of subscribers. In such a case, you might employ a password ID login scheme for every user trying to access the system. The key goal of this task is to understand who will be using the system so that architecture team can better understand user-related requirements and constraints.

[1] The use of public key cryptography to authenticate users.

Develop Prioritized Security Solution Requirements

Prioritizing the security requirements is critically important. You must define what requirements are more important and what ones are less critical. This is especially true for Internet projects in which time, perhaps the most valuable of all resources, can be very scarce. Knowing what is most critical gives you a solid foundation for tackling the next phase, architecture and design, where you put pen to paper (or mouse to keyboard), and actually start to formulate your security solution.

Architecture and Design

NOTE: Some general principles of security architecture and design follow. The specific key tasks of this phase are discussed later in this section.

But first, we discuss the upfront elimination of vulnerabilities, offer an architecture definition, discuss the use of iterative steps in architecture, and outline technical and non-technical controls.

Eliminating Vulnerabilities Upfront

The final security architecture and non-technical controls should attempt to eliminate as many of the security vulnerabilities as possible, and assuage the others as appropriate, given the threat- and asset value-based budget considerations. This is a subtle but very important point.

Some would-be system vulnerabilities can be eliminated altogether by modifying the business or IT processes. If the value of the asset and the extent of the threat warrant, eliminating a vulnerability is often more effective than developing a set of technical or non-technical controls to mitigate it. Another consideration is how to mitigate or recover from the impact of malicious or harmful activity, either deliberate or inadvertent, that can impact the business. Examples of mitigation practices are developing a recovery or failover capability to deal with Denial of Service incidents, incorporating the public relations team in the security incident response process to help "media spin" a public embarrassment, or creating backups to assuage the loss of or damage to an asset.

Architecture Defined

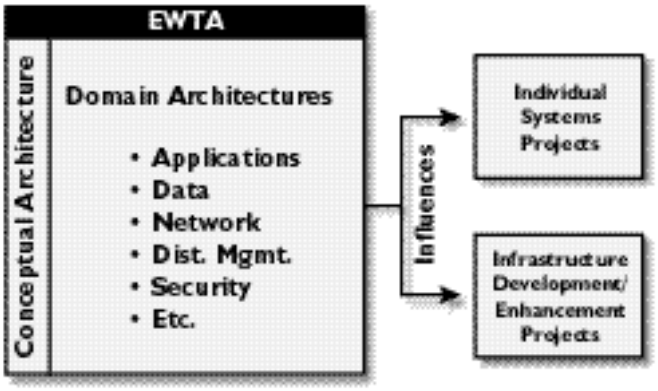
Architecture means many things to many people. For the purposes of this report, architecture is a set of principles and directions (a road map) that guides the engineering process and product selection. It includes detailed design; product selection; construction; implementation; support; and management of an organization's information systems and technology infrastructure.[2] Architecture is NOT simply an approved product list or a network diagram.

An architecture can be formulated at multiple levels. For example, an organization can define an enterprise-wide architecture that guides all development activities, including information systems and infrastructure development (networks, servers, middleware, etc.) (see Figure 5). An enterprise-wide architecture helps steer discrete projects towards a desired future state. In this context, architecture enables organizations to develop systems that meet business goals and objectives over a period of time.

Architecture also can refer more specifically to a subnetwork or a specific business system. Such a system-level architecture typically includes a more specific set of goals and requirements that drive the system design. In this discussion, we will refer to system-level security architecture as it applies to a specific business system project (see Figure 6). Our focus is on only the Information Security aspects – the technical and non-technical controls used to achieve the business security goals. In this document, we will spend the majority of time discussing the technical control mechanisms.

[2] Source: META Group, "Enterprise Architecture Strategies"

Enterprise-Wide Technical Architecture (EWTA)



Enterprise-Wide Technical Architecture (EWTA) is a logically consistent set of principles that
- Are derived from business requirements
- Guide the engineering of an organization's information systems and technology infrastructure across the various domain architectures
- Are understood and supported by senior corporate management and business units
- Take into account the full context in which the EWTA will be applied
- Enable rapid change in a company's business processes and the applications that enable them

Figure 5

System-Level Security Architecture & Design

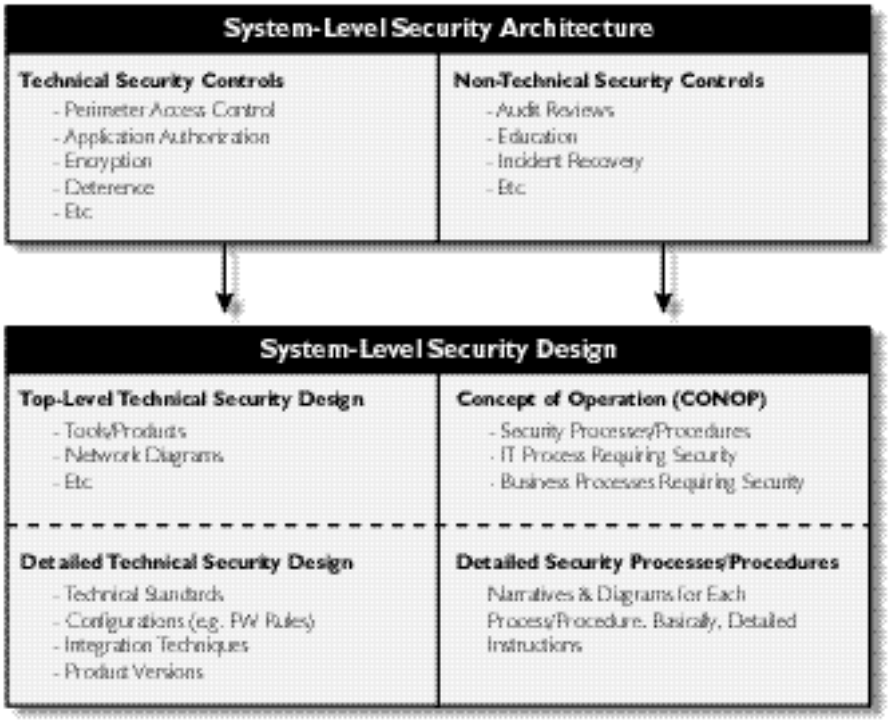


Figure 6

The line between architecture and system design is often blurry. In its pure form, which is rarely encountered "in the wild", architecture is expressed as a set of goals or requirements. Design, on the other hand, is the integration of the hardware, software, processes, and procedures required to achieve the goals. For example, security architecture may express the goal of restricting perimeter network access, but would not necessarily mandate a specific firewall, which might fall under design's purview.

Academic musing about the split between architecture and design is not important. What is critical is that you express the architecture at some level in terms of specific goals or requirements.

Iterative Steps: Practice Makes Perfect

We will walk through this phase, like the others, in a linear series of steps. However, note that finalizing the architecture and designing a security solution is often in practice an iterative process, that is, it requires several passes to get it right. The idea behind these repetitions is to catch design-level vulnerabilities that can be mitigated or effectively removed through a change in design. For example, a help desk often performs the function of resetting user passwords. This can introduce vulnerabilities into the system if the help desk employees have access to the system tools used to change the passwords. You could permit customers to reset their passwords themselves (with the appropriate authentication processing of course), or provide the help desk with only "reset" capabilities. Such steps would remove the vulnerability outright.

In the first iteration, you review the requirements set forth in the previous requirements analysis phase, along with the current enterprise-wide technical architecture or enterprise-wide security architecture and processes. With those items as input, you will begin to formulate the system-level Information Security architecture.

Architecture's Twin Components

The architecture will have two components:

1. **Technical Controls** – System controls defined in the architecture and enumerated in a system design. Technical controls might include methods such as data redundancy and tools such as Redundant Array of Independent Disks (RAID) to achieve the security goal of data availability, or token-based authentication methods like SecureID cards, to attain the confidentiality goal.

2. **Non-Technical Controls** – Controls embodied in processes and procedures. For example:

    - *Dual Entry* – The requirement of multiple signatures or approvals to allow a sensitive transaction to proceed.

    - *Separation of Duties* – The separation out of business functions so that no one individual has sole responsibility for a type of transaction.

    - *Audit Reviews* – The review of processes in an attempt to identify malfunctions or inconsistencies in operation.

    - *Awareness Programs* – Training courses or internal advertising to help modify potentially harmful end-user behaviors

System-Level Architecture Development

Formulating system-level security architecture is basically a process of inputs and outputs. In this section, we will illustrate both the traditional architecture development process, and a "fast path" method for meeting some of the stringent time requirements of the new Internet business world.

Figure 7 depicts the traditional process flow for developing a system-level security architecture. Some general principles of system-level security architecture follow. The specific tasks of this phase are discussed later in this section.

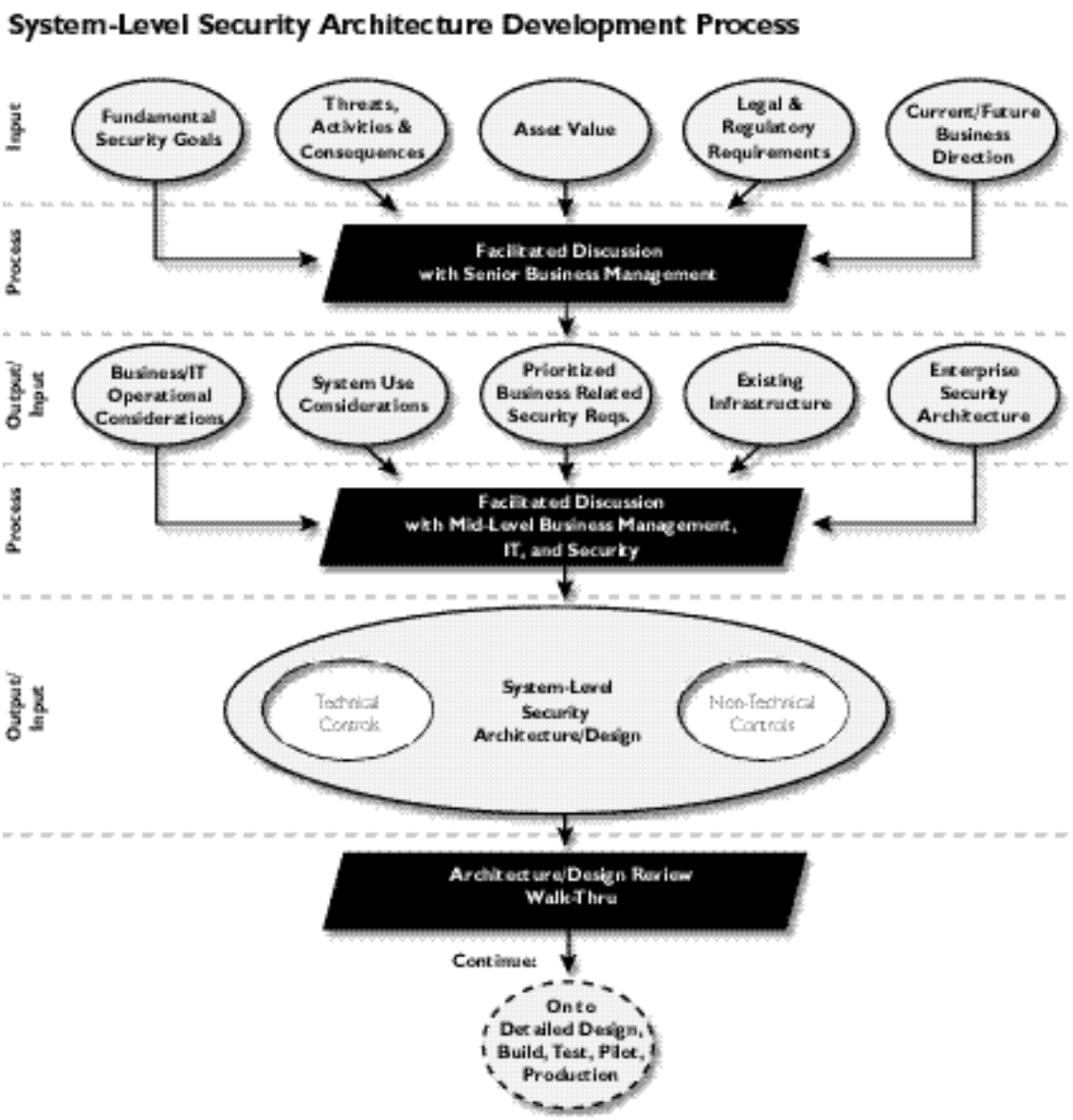


Figure 7

Key Inputs

The fundamental security goals are one of the main inputs to the system-level security architecture of a Web or e-Commerce business system. The key inputs to the system-level security architecture derive from the requirements analysis tasks above. They include:

- Security Goals
- Assets
- Threats
- Activities
- Current and Future Business Direction
- Infrastructure
- External Regulatory Requirements and Corporate Security Policy

Architecture/Design Considerations

There are two sets of architecture/design considerations:

1. First order, business-focused considerations
2. Second order, technical considerations

The output of the first set of discussions, completed in the requirements phase, will provide the grist for another series of more technical discussions that cover the second-order architectural/design considerations. These include:

- **Business and IT Operational Considerations** – Characterization of business operational and IT operational items that may impact the security architecture. For example:
  - How will new customers be set up and by whom?
  - How will users of the system be updated (e.g., password resets, changes in access rights), or removed from the system?
  - Will disparate systems be maintained, or are plans in place to further consolidate the types of systems that are operational within the IT environment?
  - Does the organization's security philosophy support central security administration and management, or is a decentralized approach preferred?
- **System Use Considerations** – Who and what will use the system, and when, where, and why will the system be used? For example:
  - Who are the end users of the system (novice or expert user, employee or non-employee, etc.)?
  - From where will users be accessing the system (home, office, company network, within the country or across country borders, etc.)?
  - When will users be accessing the system?
- **Technical Environment/Architecture** – Characterization of the current IT infrastructure that will either help or impede security.
- **Current Enterprise Security Architecture** – The current and future direction of the organization relative to security. For example, a common enterprise security architecture goal is to externalize the user authentication processing from applications to enable easier administration of user rights and privileges across multiple systems. Thus, the system architecture team needs to consider if and how this should be accomplished for the system-level security architecture. Further, many organizations are establishing a set of infrastructure security services – PKI, or proxies – that the system architecture team may want to – or be forced to – take advantage of.

These second-order discussions need to include mid-level business management as well as the applications development, IT infrastructure and security teams.

The system-level security architecture will flow out of these second-order discussions, and should include the specific prioritized goals and requirements of the architecture. At this level, the output will likely include an initial top-level design, and also should include non-technical process and procedural controls (often identified in a Concept of Operation (CONOP)).

Walkthrough Bench Test

A best practice recommendation at this point in the architecture/design process is to go engage the architecture and mid-level business management teams in a bench test – sometimes referred to as a walkthrough. The teams review the theory behind the system-level security architecture and top-level design. They apply a series of scenarios against the proposed architecture – technical and non-technical controls – to see if they achieve the desired goals prior to moving onto the detailed design.

Of course, effective security also requires sound design, construction, testing, implementation, maintenance and, in particular, training. In addition, since the new system will likely run across existing infrastructure, the architecture team will necessarily have to consider whether the current infrastructure provides adequate baseline controls for the new system. In our experience, many organizations lack appropriate technical security standards and configuration procedures that serve as the basis for a secure infrastructure. Like a house built on a shaky foundation, an application hosted on a vulnerable infrastructure is at risk. Thus, you must also include architectural requirements for the associated infrastructure, even for a system-level security architecture focused on a single application.

Baseline Best Practice Template for Internet Speed

One of the most precious business commodities in the new Internet world is speed. The ability to get to market more quickly with a business system than a competitor – "time to market" – can provide a decided competitive advantage.

Thus, the notion that "speed matters" is very appropriate to the new Internet business models. However, the traditional architecture and design process can be a lengthy one, with cycle times that are longer than many Internet business endeavors can endure. Therefore, many best practice organizations adopt new architecture development process models that improve their ability to quickly deploy new system-level architectures in general and system-level security architecture specifically. We recommend the use of best-practice base line templates. Figure 8 shows how an organization can develop and use baseline best-practice architectures to speed up the process.

The premise of this approach is that there are a relatively small number of different types of Web-based systems. Further, similar types of business systems share a set of similar information protection requirements. These requirements can be enumerated and captured as a predefined set of "best practice" system-level security architecture/design templates.

Thus, when business units decide to develop a new business system, the first hurdle for the security team is to decide what "type" of business system it will be, and to choose the template that it most closely matches. Then the first-order and even the second-order discussions can be focused on where the new business application diverges from the baseline template – this delta is often called the GAP in consulting circles. The first facilitated discussions focus on the gap between the typical security goals, threats, assets, vulnerabilities, etc. captured in the template, and the unique requirements of the application at hand. Similarly, the secondary set of discussions would use this gap information to provide the basis for a rapid development of a new system-level security architecture/design.

To use a homebuilding analogy again, this process is similar to the use of model homes. Homebuyers can pick a style, customize just a few components, and move into their development tract in no time.

Using baseline best practices also helps you achieve the legal or regulatory standard of due care in protecting company-owned assets, and assets in which the company plays a custodial role. By benchmarking the baseline templates against the industry and continuing to evolve them to meet best practice, the organization is taking prudent and responsible steps, which are typically part of the due care considerations that regulators and courts review.

Finally, do not forget to appropriately size the infrastructure to meet the overall performance goals of your applications. Performance is especially important when dealing with an Internet environment. User calls to Web servers usually place a great load on the system. Turning on the

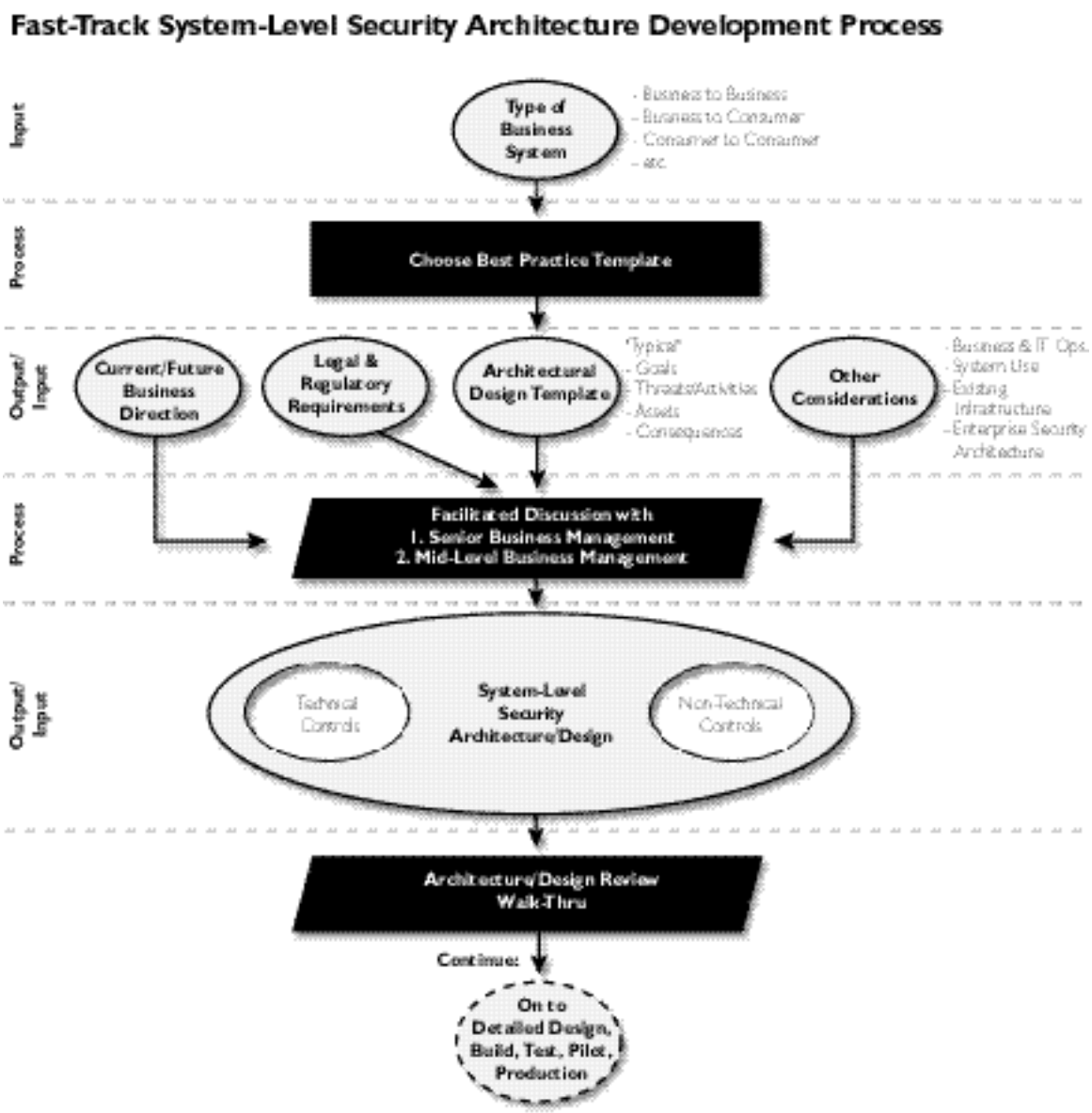


Figure 8

security features, for example, the logging services, will negatively affect system performance by taking up a lot of system cycles. Encryption services for securing data transmitted over the Web are notorious for being resource hogs. The underlying hardware infrastructure may not have been adequately sized in order to take security programs into account.

Architecture and Design Tasks

We now again pick up the discussion of key architecture and design tasks.

The following tasks are discussed:

- Create System-Level Security Architecture
- Perform Architecture Walkthrough
- Create System-Level Security Design
- Perform Design Review
- Educate Development Teams on How to Create a Secure System

- Design End-User Training Awareness Measures
- Design a Security Test Plan
- Assess and Document How to Mitigate Key Application and Infrastructure Vulnerabilities

Create System-Level Security Architecture

Key subtasks follow for the task, Create System-Level Security Architecture.

Identify Technical Security Controls
At this point, the task is to develop a list of the technical security controls that trace back to the prioritized security goals and requirements. These include technical control methods such as:

- Network, and application access controls
- Authentication requirements
- Encryption
- Redundancy
- System hardening (configuration) procedures
- Etc.

Identify Non-Technical Security Controls
Similarly, the task here is to identify a list of non-technical controls that trace back to the previously determined security goals and requirements. They include items such as:

- Multiple-signatures
- Separation of duties/functions
- Etc.

Other important tasks in the architecture and design phase follow.

Perform Architecture Walkthrough

At this stage, the key is to review the proposed technical & non-technical controls to ensure they accomplish the security goals.

Create System-Level Security Design

The design process will likely be an iterative process that will start with more basic lists and be enhanced to provide additional specificity. In the top-level design the tools, services, sub-systems, etc. that will provide the technical security controls are more fully detailed. This would include detailed security tools/service lists, top-level network diagrams, data flow diagrams, object lists, etc. On the non-technical side specific processes/procedures are listed and top-level process integration diagrams are created in the CONOP (detailed below). The processes/procedures are detailed in subsequent steps.

Top-Level Non-Technical Security Design – Concept of Operations (CONOP)
A Security Concept of Operations (CONOP) is a high-level outline of security-related processes and procedures.

Its common processes and procedures break down into the following categories:

- **Core Information Security Processes** – Address the most critical aspects of security, and are typically created, owned, maintained, and performed by the security organization, or a matrix team with security responsibility. Processes are usually of long duration (days, weeks, months) or ongoing in nature.

- **Core Information Security Procedures** – Typically created and refined by the security organization, or a matrix team with security responsibility. Procedures are typically of short duration (a few minutes or days).
- **Information Technology (IT) Processes** – Typically managed by various IT departments, but

require some level of integration with security processes and procedures.
- **Business Processes** – Maintained outside of the IT department, and require integration with various Information Security processes or procedures.

Refer to Appendix A for a comprehensive list of security-related processes and procedures.

Perform Design Review

Design reviews tend to be an iterative process. After conducting a high-level review, you work down to a detailed design review in order to fashion a final design.

The high-level review should include an examination of requirements and current architectures, design solutions, a review of vulnerabilities, and an assessment of whether and how vulnerabilities can be eliminated.

The detailed review should include a formal walkthrough. A key security-related area to check is the performance requirements and sizing of the system infrastructure (networks, servers, etc.), since security services will often overload services.

It is important to go through a set of design reviews focused on the security aspects of the system. Depending on the design area being reviewed, business, application development, and security personnel should take part in these reviews.

The series of design reviews should include the following:

- Technical review geared to the application level
- Technical review geared to the infrastructure level
- Non-technical review (processes and procedures)

Educate Development Teams on How to Create a Secure System

The two main aspects of this task are:

- Provide best practices for secure coding.

- Provide practical education on using the various security tools and services.

Application teams are, naturally, usually well-schooled in the art of development. However, they frequently do not understand the security aspects of development. The application teams require education and training in such matters as secure coding practices. They need to learn how to identify common vulnerabilities, and how to establish secure development environments. In addition, developers should be exposed, like the rest of the organization, to the general security principles.

Another aspect of this task relates to training developers in specific security tools and services. In newer systems, developers make frequent use of external security services by means of application program interface (API) calls. In the "good old days" of mainframe "legacy system coding", many security-related functions like authentication and auditing were done within the application. In modern Web environments, by contrast, many security functions like access control are done via middleware services. In such settings, the developers can use within the operating environment a set of third-party tools, such as PKI, or services layered into operating systems, such as logging, auditing, authentication, etc. to accomplish security functions.

Design End-User Training Awareness Measures

The design of a training and awareness program is vitally important. All the security processes and hardware in the world will do little good if your personnel does not know how to use them. This is often missed or overlooked in training manuals and courses. In Web applications, especially with external users, security training and awareness often must be built into the application, or

incorporated in "getting started" type of user education tools.

Design a Security Test Plan

You should perform a basic check of the components of the system test plan. Security needs to be wrapped into the overall test plan. It should be communicated to the integration and deployment teams.

In addition to traditional systems testing, you want to focus on certain specific areas. These include:

- **Usability/Ergonomics Testing** – Checks that security features function, and are reasonably well understood and easy to use. If the security features do not have these characteristics, they will be ignored or will cause problems. Usability/ergonomics is typically applied to applications themselves, but can be applied to security-related matters as well.

- **Performance Testing** – Checks that the system efficiency performs as required. This is traditionally done as part of the regular SDLC. However, as mentioned above, testing after turning on the security-related features – authentication, encryption, etc. – is important in determining if system performance is adequate. The impact of security features on performance is often not noticed until after the system is put into production.

- **Vulnerability Testing** – Also known as penetration testing or "ethical hacking", attempts to uncover critical vulnerabilities so that they can be fixed before production. Because vulnerability testing is not part of the traditional SDLC, it is often overlooked.

Assess and Document How to Mitigate Key Application and Infrastructure Vulnerabilities

This task is a variation of bug tracking, in which you highlight security-related deficiencies. You have several options for handling these flaws. You can remove or mitigate them by tuning the design or reconfiguring the infrastructure. Alternately, if you are unable to eliminate or lessen vulnerabilities, you can deter them, or at least monitor for their occurrence.

A parking lot owner wishing to reduce car thefts, for example, might deter thieves with extra street lighting, and monitor suspicious activity with surveillance cameras. In the Internet world, you could deter intruders by posting warning banners or copyright notices on your Web site. Some vulnerabilities may prove impossible to fix, and others prohibitively expensive to address. You definitely want to monitor for such vulnerabilities. You might watch for would-be crackers through, for example, intrusion detection products that detect tell-tale signs of attacks.

Develop (Build/Configure/Integrate)

The development phase implements the requirements and architecture/design decisions.

Your security goals can be applied during the development phase. With some high-risk systems, a secure development environment is set up during this phase. This would include the proper policies and procedures to protect the integrity and confidentiality of the code, the designs, and underlying development infrastructure from related attacks such as theft or destruction.

The essential part of development is coding, that is, the writing of the actual programming code. Some companies have coding standards – specific rules for code structure and appearance. Standards can prolong the life of the code, and make it easier to maintain. Writing secure code from the start saves time and money during testing and the remainder of the life cycle.

Many organizations are adopting secure coding standards to ensure that the code is secure. To achieve this aim, developers should use defensive programming techniques.

Examples of security-related development considerations are:

- Identify, in terms of security, the weaknesses and strengths of the programming language.
- Put system source code and related configuration files in a secure environment, preferably one isolated from the rest of the network.
- Set up a code/component version control system.
- Develop operational procedures (based on the CONOP). Ensure that network and systems administrators, partners, customers, and integration firms that deploy the solution or solution components meet the requirements outlined in the technical standards for the mitigation of vulnerabilities.

Defensive techniques are further detailed in the METASeS publication *Building Secure e-Commerce Applications*.

Develop Technical Configuration Standards and Procedures

Especially in the case of large system rollouts, it is likely that the different teams will roll out the application and the underlying infrastructure.

You should document how to perform these rollout activities securely, in a set of publications that outline: 1) technical security standards; and 2) technical configuration procedures.

Technical security standards provide detailed rules that define what should be done at a technology level to mitigate security risks. The standards are related to network or systems infrastructure, for example, a router, operating system, network switch, or server. Other technical standards are related to the business applications being rolled out.

Technical configuration procedures provide step-by-step instructions on how to configure an operating system or service, or a security software tool, such that it adheres to the rules detailed in the technical standard. For example, there may be step-by-step configuration instructions on how to securely configure a Windows NT server, or the newly developed application, to comply with security standards.

Such publications help ensure that personnel are properly hardening the system by removing known vulnerabilities such as default passwords or unneeded services. (METASeS offers a full suite of *Security Standards* and *Configuration Procedures*.)

Test Security

The test phase verifies that all of the security-related components of the system work as advertised, and meet the expectations of owners and end users. In most cases, the security aspects of testing are integrated with the regular test plan. The exception is vulnerability testing.

The first two tasks are typical test functions, with a security twist. The third task – vulnerability assessment – is a recently developed test function that is specific to security.

Conduct Performance/Load Test With Security Features Turned On

This task involves verifying that the system performs up to specification and meets production load requirements with the security components and features that are engaged.

This task is often overlooked. Do not make the elementary mistake of forgetting to turn on all of the security features, discussed above, that apply to your system.

Perform Usability Testing of Applications That Have Security Controls

You should verify that security-related functions are ergonomically sound. If personnel are not accomplishing their work through the applications, for example, because the applications are overly time-consuming or difficult to use, you have to take remedial steps like training the employees or reengineering the system.

Perform Independent Vulnerability Assessment of System (Infrastructure and Applications)

You must verify that the infrastructure and applications cannot be compromised, and that attackers cannot use the applications to carry out unintended functions. Verification is essential prior to an application being implemented on your system or shipped to a customer, before a security flaw can do any damage.

Vulnerability assessment has traditionally examined the infrastructure through such "ethical hacking" means as penetration testing. Still, you must not neglect assessing applications as well, especially in an Internet environment, where a skillful cracker can manipulate Web applications into performing unintended functions that compromise data. Crackers put large amounts of data into fields intended for only a few bytes. This technique, called a buffer overflow attack, can enable an attacker to directly control a system, or take control later by providing commands as part of the input string. One of the methods of accomplishing partial application-level vulnerability testing is to have the functional testing team add a set of negative test cases. Negative test cases refer to checking various functions, input fields, etc. for the result of unintended inputs. This is the opposite of positive test cases where the functional testing team is expecting certain results based on various application inputs. As outlined above, hackers will often try things that the application would not expect – e.g., buffer overflow situations. (For an intensive review of vulnerabilities to Web applications, refer to the pathbreaking METASeS report, *Building Secure e-Commerce Applications*.)

This verification is best performed by an independent organization. This would be an internal auditing group not associated with the application development effort, or an outside firm. The development teams have typically worked too closely with the products to objectively assess their security vulnerabilities.

> ***NOTE:*** A number of companies, including METASeS, specialize in performing independent vulnerability assessments.

Independent, third-party verification can help to protect your organization against the threat of legal liability in the event of security incidents. Vulnerability assessments indicate, in legal parlance, that your organization is exhibiting "due care" to protect its own assets and that of clients. The courts recognize that there is no such thing as "perfect security." However, through the doctrine of due care, they do place the burden of taking prudent measures to protect assets upon the organization responsible for creating and maintaining the infrastructure and applications.

Deployment/Implementation

This phase involves taking an application from the testing environment to the production environment for limited (e.g., pilot) or full-production uses.

Simply put, you want to make sure the people carrying out the deployment follow the security-related processes and procedures that you set up in the preceding phases.

Deploy Training and Awareness Program

This task entails implementation of a security training and awareness program. Administrative personnel and users should be schooled in the system's security functions.

Operations/Maintenance

After the system is deployed, you continue to operate and maintain it. From a security perspective, operations and maintenance may include:

- Test and migrate to new software versions. Especially important because vulnerability conditions change over time. Patches and service packs will address known vulnerabilities.
- Conduct periodic risk review and consult with the business/application owner to assess whether risks have changed. Make appropriate upgrades or downgrades.
- Continue to provide as well as strengthen the security awareness program, reminding end users of the things they need to do to maintain security. This is important because new users are added and new vulnerabilities arise continually, and existing users tend to let down their guard.
- Conduct periodic vulnerability assessments. Be aware that hackers are discovering new vulnerabilities and devising new threats all the time. It is critical to catch vulnerabilities in a new release before the application is made public. You should perform quarterly to yearly vulnerability assessments, based on the risk profile of applications. It is particularly important to perform vulnerability assessments with new releases of the system, as this is often when new vulnerabilities arise.
- Perform auditing, logging, monitoring, archiving. This should include periodic audits of processes and procedures, as well as ongoing review of logs, especially when monitoring for known weaknesses.

Perform Ongoing Vulnerability Monitoring

Crackers continually develop new tools and techniques for exploiting vulnerabilities. You need a process for the continual monitoring of new vulnerabilities. A number of organizations, such as Bugtraq, CERT, and METASeS – through its Web Alert service – provide a service that keeps you informed of the latest security vulnerabilities.

Because you will not be able to eliminate all vulnerabilities, you will want to regularly monitor holes in your defenses. This is typically done through an Intrusion Detection System (IDS). (METASeS conducts "ethical hacking" of clients through an IDS to probe for vulnerabilities.) Because it is impossible to close every hole, a key ability is to react swiftly and effectively in the event of a security incident.

For example, a Denial of Service (DoS) attack is nearly impossible to prevent. However, you can monitor for DoS attacks, engineer system redundancies and failovers to mitigate the negative impact of a DoS, and have a plan of action ready in case one occurs.

> ***NOTE:*** Later versions of this report will discuss packaged applications, including those provided by application service providers, and those purchased commercially off-the-shelf (COTS), for example, SAP and PeopleSoft.

Conclusion

Ensuring system security can seem an overwhelming job, given the complexity of technologies, the wealth of new applications continually entering the marketplace, and the growing number of attacks on corporations from external intruders.

You will never to able to provide perfect security for your organization. However, you can significantly improve security, and save a great deal of time and effort along the way, by marrying security to every phase of the SDLC.

Appendix A -

Checklist of Security-related Tasks & Subtasks

✓	Phases/Tasks	Why	Who	Task Output (What)
✓	**Requirements Analysis**			
	Analyze security requirements	Ensure security-related requirements are factored into the system architecture & design.	Business Execs, Application Development Mgrs., Security Team	Prioritized security-related requirements (including the documentation from the sub tasks below)
	-- Evaluate security risks, and consequences.	Educate business team, understand key sensitivities & business consequences of various breach scenarios.	""	None (done for educational purposes)
	-- Perform information asset value analysis.	Understand the value of the assets that need to be protected so that the solution (technical and non-technical security controls encapsulated in system-level security architecture/design) are appropriate for the value of the asset.	""	A document identifying critical business assets & protections required.
	-- Discuss potential threats.	Understand who the actual threats would be so that security controls can be prioritized.	""	Prioritized list of threats
	-- Analyze potentially malicious or harmful activities.	Understand the types of security-related activities threats can engage in so that the controls can address them.	""	Prioritized list of potential security activities
	-- Analyze high-level vulnerabilities.	Understand the types of vulnerabilities that need to be addressed.	""	Prioritized classes of vulnerabilities
	-- Discuss security goals (confidentiality, integrity, availability, non-repudiation, audibility).	Understand key security requirements/goals required to mitigate key business risks and consequences.	""	Document with key security related design goals
	-- Review regulatory requirements and corporate Information Security policy.	Understand regulatory requirements and existing security policy.	""	Document listing regulatory requirements and applicable security policy

	Phases/Tasks	Why	Who	Task Output (What)
	-- Review future business goals.	Understand future business goals for the system to ensure the proposed solution will be extensible to future requirements. All potential future requirements cannot be known, but at least understanding likely future requirements will enable the team to avoid architecture/design dead ends.	""	List of likely future business requirements
	Discuss business/IT operations.	Understand the context that the new system will operate in to ensure it meets those operational requirements.	Business & IT operational managers, Application Development, Security Team	Notes regarding how the system will function within the business and IT environment. Who will do what, when, etc. This list is often a part of the traditional lifecycle output. The details are not needed at this point
	Review current security program (policy, procedures, standards, processes, technology).	Understand current security capabilities vs. requirements for the new application. Highlight any deficiencies in the current security program so they can be addressed either separately from the system project, or by the system-level security architecture	Security Team	Report - Security Program Gap Analysis. Analysis of the current Information Security policy, standards, procedures, processes, technology.
	Profile users for the systems.	Understand key project scope issues and design considerations relating to who will be using the system (who, when, where, why, etc.)	Business & IT operational managers, Application Development	List of users (who, what, when, where, why)
	Understand customer partner interface requirements (business-level, network, etc.)	Understand architecture/design considerations & constraints.	Business Team, Application Dev., Security Team, Infrastructure Teams	List of internal and external systems/networks where integration is required. Should begin to document some of the technology involved, but detail not required at this point (e.g., TCP/IP network, XML and HTTP, Oracle DB, etc.)
	Discuss project timeframe.	Understand time constraints	Business & IT operational managers, Application Development, Security Team	Top-level project plan/milestones
	Develop prioritized security solution requirements.	Document business- and technology-related security requirements & scope for use in the architecture/design phase	Business & IT operational managers, Application Development, Security Team\	High-level requirements document (narrative text & bullet points with appendix of related reference material)
	Decide cost & budget constraints for security solution (development & operations).	Understand budget constraints and develop a rough order of magnitude for security-related budget. It is very important to cover not only the upfront development costs, but also the ongoing operational costs (e.g., ongoing vulnerability monitoring and assessments, ongoing threat monitoring/log file review, etc.)	Business Team Security Exec Application Dev. Exec	Security-related budget (initial development & operational budget)

	Phases/Tasks	Why	Who	Task Output (What)
	Sign off on security requirements and budget.	Gain approval for documented requirements & rough order of magnitude budget.	Project Sponsors & Project Managers	Appropriate approval signatures
	Decide on buy vs. build for security services.	Need to determine who can execute the architecture/design, development, testing, implementation, and operational security tasks (in-house staff, external staff or some combination)	Security Team, Infrastructure Teams, Application Development Team	Buy vs. build for various security-related tasks
	Architecture and Design			
	Create system-level security architecture. Includes:	Document the requirements and goals for the security solution needs.	Application Development, Security Team, Infrastructure Team	High-level statement of requirements accompanied by various technical methods of achieving the requirements (e.g., a network access control on the technical side, or audit reviews on the non-technical end). The architecture should focus on the what and why rather than the how. Some design items will be obvious, e.g., a firewall to address network access controls. However, the key at this point is not to get into detail design! It is important to remain high level enough to be able to review and adjust the architecture prior to expending too much time on the design.
	-- Technical security controls	To document the technical controls necessary to achieve the requirements	""	List of technical controls needed (e.g., access control, authentication, encryption, deterrence measure, etc.)
	-- Non-technical security controls.	To document the non-technical controls (processes, procedures) necessary to achieve the requirements	""	List of non-technical controls needed (see core security processes & procedures in appendix C)
	Perform architecture walkthrough.	Ensure that proposed controls will adequately and effectively meet the requirements	Mid-level Business Team, Business Application Development, Security Team, Infrastructure Team	List of deficiencies that need to be corrected in the architecture
	Create system-level security design. Includes:	Document the specific technical and non-technical controls that will be used.	Security Team, Applications Development, Infrastructure Teams	Document the specific techniques - how the goals/requirements laid out in the system-level security architecture will be achieved (e.g., D27selecting specific methods and tools for achieving the goals)

	Phases/Tasks	Why	Who	Task Output (What)
	-- Top-level technical security design	Document more specifically what technical security elements are needed and how they will achieve the requirements	Security Team, Applications Development, Infrastructure Teams	List of tools and products that will be part of the technical solution and how they will work with other elements (other technical security controls, the application system itself, infrastructure). The output will likely be in the form of a document or presentation with a combination of narrative text and graphics (e.g., network diagrams).
	-- Top-level non-technical security design (CONOP)	Develop a security CONOP (Concept of Operations), a high-level outline of the required processes and procedures and their interrelation.	Security Team, Applications Development, Infrastructure Teams	Narrative document with top-level process flow charts and procedure definitions
	-- Detailed technical security design.	To add additional specificity to the top-level technical design.	Security Team, Applications Development, Infrastructure Teams	The detailed design will articulate specific vendor tools, versions, integration code, configuration setting, rule sets, etc.
	-- Detailed security processes and procedures	To add additional specificity to the top-level non-technical design.	Security Team, Applications Development, Infrastructure Teams	The detail design will document the specific process and procedural steps for each of the processes identified in the CONOP. This includes specific technical security standards and configuration procedures for appropriately hardening the system (hardware, operating software, middleware, application, etc.)
	Perform cost/benefit analysis for various design elements.	This task should be done prior to the detailed design. The purpose is to evaluate various design scenarios and make cost/benefit trade-offs to establish a final top-level design.	Security Team, Applications Development, Infrastructure Teams	Cost & Benefits comparison chart for various design alternatives. Priority given to the solution(s) that most closely meet cost/benefit and security architecture goals
	Perform design review. Should include: technical review geared to the application level, technical review geared to the infrastructure level.	Catch and fix any design flaws in the system. These reviews are typically done in an iterative fashion after each level of design is complete. The purpose of the more detailed reviews is to understand vendor/tool integration, performance issues, and make product selections that meet architecture requirements. Top-level design reviews are typically done with design walk-through sessions, while more detailed reviews sometimes include lab testing for some of the security infrastructure	Security Team, Applications Development, Infrastructure Teams	Prioritized list of required design changes
	Educate development teams on how to create a secure system.	Ensure the development teams are appropriately educated about general security issues, development-related security issues (e.g., secure development environments) and the security related tools they will need to utilize	Security Team, Applications Development, Infrastructure Teams	No output (educational purposes)

	Phases/Tasks	Why	Who	Task Output (What)
	Design end-user training and awareness programs.	Ensure the end users of the system understand how to use the security features correctly and how not to do adversely impact security (use strong passwords, don't share passwords or credentials, etc.)	Security Team, Applications Development, Business Units	Training and awareness documentation, system documentation, help systems, installation procedures, videos, awareness aides built into the system, etc.)
	Design security test plan.	Document how to test the solution (performance, vulnerability, code reviews, assurance, etc.)	Security Team, Applications Development, Infrastructure Teams	
	Update Information Security policy, if appropriate. Updates should NOT be performed often.	Some new applications will require a reevaluation of current Information Security policy.	Security Team	Additional policies, or alterations to existing policy. This should be the exception rather than the rule and not done very often if the policy is written correctly.
	Assess and document how to mitigate key application & infrastructure vulnerabilities.	To document and prioritize vulnerabilities and fixes, not only for the infrastructure and middleware components, but also for purchased packaged applications	Security Team	Database of vulnerabilities that enables the tracking of those that are addressed and those that are still outstanding.
	Design security for development & test environments.	To ensure the development and test environments are secure (source code version control procedures).	Security Team & Application Development Team, Infrastructure Team	Processes, procedures and technology (e.g., version control systems, access controls, segregated development/testing networks, etc.)
	Develop (Build/Configure/ Integrate)			
	Set up secure development environment (e.g., development servers).	Ensure control & integrity of the source code & component configurations	Security Team & Application Development Team, Infrastructure Team	Secure development environment
	Train developers on security-related middleware and secure coding practices (authorization services, encryption, PKI, etc.)	Ensure the development teams are appropriately educated about general security issues, developmen- related security issues (e.g., secure development environments) and the security-related tools they will need to utilize.	Security Team & Application Development Team	Trained and educated developers
	Train infrastructure teams on installation & configuration of the middleware.	Ensure the infrastructure teams understand how to use & configure the new infrastructure required for the system.	Security Team, Infrastructure Team	Trained and educated Infrastructure Team
	Code application-level security components.	Develop code per design.	Application Development	Application code
	Install/Configure/Integrate Infrastructure	Provide a ready technical security environment for various development and testing.	Security Team, Infrastructure Team	Ready infrastructure (HW, OS, middleware services, etc.)

	Phases/Tasks	Why	Who	Task Output (What)
	Set up security-related vulnerability tracking process.	Ensure identified vulnerabilities are systematically recorded, prioritized, and appropriately removed	Security Team & Application Development Team	Tracking database (could be a simple list/spreadsheet). Often integrated into the QA or bug tracking system. It will be necessary to code security-related issues for easy recognition and prioritization (sorting, reporting, etc.)
	Develop detailed security test plan for current and future versions.	Ensure the security-related testing is done.	Security Team & Application Development Team, Infrastructure Team	Documented test plans for security-related testing
	Conduct unit testing & integration testing.	Ensure the security-related services being used by the developers are working (e.g., external authentication or encryption services, network security services like VPN's, etc.). This is typically an iterative process where various developers are testing specific code components or subsystems as they develop the codes.	Security Team & Application Development Team, Infrastructure Team	
	Test			
	Perform code review/code walk-through.	Verify application code meets security requirements/specifications	Security Team & Application Development Team, Infrastructure Team, Independent Audit team	List of code-level security issues to be corrected
	Test the configuration procedures.	Verify that administrators can use the procedures to set up network and systems components to meet the standards, and that vulnerabilities are in fact removed or mitigated.	System deployment teams	List of procedural issues that need to be resolved/clarified
	Perform systems test.	Verify technical components are functioning together to design/architecture specifications	Security Team & Application Development Team, Infrastructure Team, QA Teams	List of system-level and system integration-related issues that need to be resolved
	Conduct performance/load test with security features turned on.	Verify systems performance & production load requirements with security components/features running	Security Team & Application Development Team, Infrastructure Team	List of security-related performance issues that need to be resolved
	Perform usability testing of applications with security controls.	Verify application with security controls meets customer usability/ergonomic needs and make adjustments.	Application Development Team, QA team.	List of security-related usability issues that need to be resolved

	Phases/Tasks	Why	Who	Task Output (What)
	Conduct independent vulnerability assessment of the system (infrastructure & application).	Independent verification to ensure the infrastructure and application cannot be compromised	External Auditor (either within the organization or 3rd party)	Prioritized list of vulnerabilities
	Deployment (Pilot, Full-Scale Deployment)			
	Conduct pilot solution (infrastructure, application, etc.)	Make deployment & operational process adjustments prior to full-scale implementation.	Security Team & Application Development Team, Infrastructure Team	Prioritized list of pilot-related issues to resolve
	Conduct transition between final system test and development through secure means such as change management procedures and encrypted transmissions.	Ensure the integrity of the final system.	Application Development Team, Infrastructure Team, Change Management	Change Management report/sign-offs
	Ensure that system files are compared to originals to ensure authenticity.	Ensure the integrity of the final system	Development Team	Issues if any
	Deploy training and awareness program. Train administrative personnel and users in the system's security functions.	Ensure end users are appropriately trained on the security-related system features, processes, and procedures	Development Team	Educated and trained end users
	Conduct full-scale deployment.	Need to make sure current infrastructure is secure before installing new system.	Development Team	Change Management report/sign-offs
	Operations/ Maintenance			
	Test and migrate to new software versions	Especially important because vulnerability conditions change over time.	Security Team & Application Development Team, Infrastructure Team	Updated vulnerability punchdown list.
	Conduct periodic risk review.	Business risk conditions change over time and need to be reassessed. Consult with the business/application owner to assess whether risks have changed.	Security & Business Teams	Risk review report
	Continue to provide as well as strengthen the security awareness program	New users are added and new vulnerabilities arise continually.	Security Team	Improved security awareness program, and prioritized security awareness initiatives
	Conduct periodic vulnerability assessments	Existing users tend to let down their guard.	Security Team	Vulnerability reports

	Phases/Tasks	Why	Who	Task Output (What)
	Perform auditing, logging, monitoring, archiving.	Hackers are discovering new vulnerabilities and devising new threats all the time. It is important to perform vulnerability assessments with new releases of the system, and catch vulnerabilities in new releases before the application is made public. Need to ensure possible breaches are recognized and addressed. Audit processes and procedures and review logs, especially for known weaknesses.	Security Team & Various Operations Teams	Status reports

Appendix B -

SDLC Overlay

"Typical" SDLC Phases	Microsoft Solutions Framework (MSF)
Requirements Analysis	**Envisioning Phase** - Vision Approved Milestone with Following Output: Vision Document, Master Risk Assessment Document, Project Structure Document
Architecture and Design	**Planning Phase** - Project Plan Approved Milestone with Following Output: Functional Specificity, Master Project Plan, Master Project Schedule, Updated Master Risk Assessment Document
Develop (Build/Configure/Integrate)	**Developing Phase** - Scope Complete Milestone with Following Output: Complete Functional Specification, Updated Master Project Plan/Schedule, Updated Master Risk Assessment Document, Initial Performance Support Elements, Test Spec, and Test Cases.
Test	**Stabilizing Phase** - Release Milestone with the Following Deliverables: Golden Release, Release Notes, Performance Support Elements, Test Results, and Testing Tools
Deployment (Pilot, Full-Scale Deployment)	**Note:** MSF is oriented toward building software the others will deploy
Operations/Maintenance	**Note:** MSF is oriented toward building software the others will deploy.

Appendix C -

Security Related Processes & Procedures

- Core Security Processes
- Information Security Procedures
- IT Processes
- Business Processes

Core Security Processes

Core security processes are typically owned (created and refined) and performed by the security organization, or a matrix team with security responsibility.

Process	Definition
Security Program Management	The ongoing management, review and refinement of the elements of the Information Security program, including policies, standards, procedures, processes, technology, and organization. Addresses changing risk and business conditions.
Risk Management	The ongoing, methodical assessment of Information Security risks across the organization.
Policy Management	An ongoing (lifecycle) process for defining and refining the organization's Information Security policy framework elements (policy, procedures, standards).
Threat Management	The ongoing monitoring (e.g., open source data collection) and management (deterrence, mitigation, tracking, etc.) of Information Security related threats. Treats are would-be perpetrators (people/organizations) of Information Security breaches.
Vulnerability Management	The ongoing monitoring and management of Information Security vulnerabilities. Vulnerabilities are the holes threats could exploit to breach of Information Security.
Security Administration	Adding, changing, user, group, or system privileges.

Process	Definition
Security Architecture	Development and ongoing maintenance of the security architecture -- typically a sub architecture or domain of the enterprise-wide technical architecture. This process may also refer to the development and maintenance of a portfolio of system-level architecture/design templates enabling rapid deployment of new systems through the use of baseline best practice templates.
Information Security Training & Awareness	Development an ongoing maintenance of the security education program. This includes awareness activities oriented towards the end users (employees, customers, partners, etc.), as well as training on Information Security policy, processes, architectures, procedures, etc. for the IT organization (e.g., applications development, operations).

Information Security Procedures

Information Security procedures are typically created and refined by the security organization or a matrix team with security responsibility. Procedures are differentiated from processes by their scope and duration. Procedures are typically of short duration (a few minutes of days), while processes are more often long-running (days, weeks, months) or continuous in nature.

Information Security Procedure	Definition
Technical Configuration Procedures (a.k.a. System Hardening Procedures).	Specific instructions on how to configure various systems (network & system devices, applications, etc.) such that they comply with the organization's Security Standards. Organizations typically require a portfolio of Technical Configuration Procedures to address the multiple systems in use.
Media Sanitation	The procedure for disposing of information that contains sensitive data.
Vulnerability Assessments (a.k.a. penetration testing, ethical hacking)	The instructions on what and how to test an Information Technology environment and uncover vulnerabilities (both system and non-system level).
Incident Response	Instructions to handle potential or real security breaches. The procedure typically includes an incident prioritization method, escalation instructions and matrix of resources to involve based on the type and severity of the incident.
Incident Investigation/Forensics	The procedures for researching the source of a breach and gathering evidentiary data in support of potential future civil or criminal litigation.
Threat Assessment	The procedures for analyzing specific threats to an organizational asset (or set of assets).
Asset Classification	Instructions/guidance that information/data owners can use to appropriately value and label the information/data in question, such that it receives the appropriately secure treatment (both via technical and non-technical controls). Part of asset classification is system zoning, formal zones that identify infrastructure grouping by priority. Corresponds to the level of monitoring implemented for certain systems.

IT (Information Technology) Processes

IT (Information Technology) processes are typically managed by various IT departments, but require some level of integration with security processes and procedures.

Process Relationship to Information Security	Process Definition
Configuration Management Information Security Relationship -- Vulnerability management process and vulnerability assessment procedures to ensure known problems are fixed (e.g.,. via patches or service packs) and new problems are not introduced to the environment.	The process for making modifications to systems including: hardware, operating systems, network operating systems, application systems, etc. The configuration management process typically also endeavors to document the physical and logical relationships and specific configuration of system elements.
Change Management Information Security Relationship -- Vulnerability management process to ensure that appropriate security sign-offs are completed so that new vulnerabilities are not introduced to the environment.	The communication, tracking, and approval of modifications to the production environment for new systems or various configuration modifications, such that they are performed in an orderly, controlled manner to minimize adverse impacts and enable rapid recovery in the event of a problem. Broad categories of change may affect certain systems, e.g., programmatic changes, system updates (patches), configuration changes (OS or middleware). Each change may activate high-level responses. System updates may require that a system be recertified within certain days of the change. Significant changes to middleware (WWW server) applications require manual examination by Information Security.
Contingency Planning/Disaster Recovery Information Security Relationship -- Ongoing threat management process and security incident response procedures to ensure the cross communication and orderly restoration of systems in the event of a security breach. Also the risk assessment process so that the proposed contingency/disaster recover plan does not introduce security vulnerabilities (e.g., offsite storage of sensitive backup media may require additional controls).	Process designed to minimize the impact of adverse events and ensure an orderly restoration of the IT capability supporting the business. The adverse events can derive from either man- made and natural disasters, and may stem from accidental or purposeful acts. They include natural disasters, or intentional acts like sabotage or security breaches (e.g., Denial of Service attacks).
Enterprise-Wide IT Architecture Information Security Relationship -- Security architecture is a subarchitecture (or component architecture) to the enterprise-wide IT architecture.	An enterprise-wide technical architecture (EWTA) is a logically consistent set of principles that guides the "engineering" -- that is, detailed design, selection, construction, implementation, deployment, support, and management of an organization's information systems and technology infrastructure. Consequently, a set of product standards, by itself, does not constitute architecture. What is most important is guidance on how the products are to be used to achieve the enterprise's goals.
Systems Development Lifecycle Information Security Relationship -- Security architecture, risk assessment, vulnerability assessments, security administration. The SDLC needs to incorporate various security tasks so that appropriate security is built into the systems rather than bolted on afterwards.	The process that governs how systems are built, or modified to meet the organization's needs. This includes COTS (commercial off the shelf) systems that can be purchased and deployed or purchased and modified/integrated to fulfill requirements.
IT operations (a.k.a. data center operations or network operations (NOC)) Information Security Relationship -- Threat management, incident response. The NOC or data center operations staff often provides off-hours threat event monitoring and initial incident response triage steps.	The ongoing monitoring of systems to ensure that potential problems are recognized and fixed prior to when problems or outages would be identified and addressed within defined service level targets.
Help Desk/Problem Management Information Security Relationship – Threat management, incident response, security administration (e.g., for password resets).	The process for tracking customer calls and service outages such that issues are resolved within service level parameters.

Business Processes

Business processes are maintained outside of the IT department, and require integration with various Information Security processes or procedures.

Process Relationship to Information Security	Process Definition
Employee Lifecycle Information Security Relationship -- Security administration (add, change, or remove system privileges).	The HR processes/procedures dealing with the hiring, training, promotion/demotion, or removal of employees.
Merger & Acquisition (M&A) and Divestiture Information Security Relationship -- Risk management, risk assessment (as part of due diligence), security administration, security architecture.	The business process for assessing (due diligence) and then executing mergers/acquisitions, or divestitures.
Regulatory Compliance Management Information Security Relationship -- Risk assessment	The process for maintaining compliance with regulations and laws administered by such agencies as HIPPA and the SEC. While the legal department of an organization has responsibility for this matter, Information Security must be aware of areas that impact it. In banking, for example, regulatory boards can impose fines or sanctions for technical non-compliance.
Customer Processes Information Security Relationship -- Security administration	Processes designed to add, delete, or change customers.
Partner Processes Information Security Relationship -- Security administration, security architecture (for system level integration), vulnerability assessment	Processes designed for adding, deleting, or changing partners (suppliers, distributors, resellers, etc.)
Audit Information Security Relationship -- Risk management, risk assessment, security administration	Process for review of financial and other official business operations.
Public Relations Information Security Relationship -- Incident response (for "spin control" of breaches).	Process for orderly dissemination of information relating to the organization
Product Development Information Security Relationship -- Risk assessment, security architecture (where new products contain an IT component)	The process for creating new hard goods or service products for sale.
Certification Information Security Relationship -- Risk management, risk assessment, security administration, vulnerability assessment	A formal process of signing off on the functionality and security of new systems.
Physical Security Information Security Relationship -- Physical access controls to IT infrastructure and systems	The process (or organization) responsible for securing the physical assets of the organization.