Microsoft

SDL and HIPAA

Aligning Microsoft SDL Security Practices with the HIPAA Security Rule

Updated November 4, 2010

For the latest information, please see http://www.microsoft.com/sdl

Table of Contents

EXECUTIVE SUMMARY

This paper shows how the Microsoft Security Development Lifecycle (SDL) can help meet some of the requirements of the Health Insurance Portability and Accountability Act (HIPAA). It is important for the reader to realize that HIPAA was enacted in 1996 when most physician offices operated on a paper records system and before software, software security, or software security requirements were considered a primary concern. Today, some 14 years after the enactment of the original HIPAA statute the healthcare industry is highly dependent on software. Healthcare and security professionals increasingly recognize that building security into software is critical to ensuring the security and privacy of patient records held in the software.

This paper addresses two primary scenarios—the development of new healthcare software and the integration of healthcare software into an electronic health record—where software security should intersect with HIPAA requirements. Our goal is to show where software security can both assist in attaining regulatory compliance with HIPAA and ensure that the software being created for the healthcare industry is written and deployed with security as a priority, using the SDL as a guide. Additionally, this paper highlights some HIPAA security requirements (called ***safeguards***) and demonstrates how SDL practices can be used to support those safeguards.

INTRODUCTION

The purpose of this paper is to describe how the Microsoft Security Development Lifecycle (SDL) can help organizations comply with some requirements of the administrative simplification provision of the Health Insurance Portability and Accountability Act and its implementing regulations (HIPAA), including the Security Standards for Protecting Electronic Protected Health Information (HIPAA Security Rule) and the Standards for Privacy of Individually Identifiable Health Information (Privacy Rule), as well as the American Recovery and Reinvestment Act of 2009 (ARRA), particularly Title XIII of ARRA, called the Health Information Technology (HIT) for Economic and Clinical Health (HITECH) Act. This paper attempts to present how SDL practices and HIPAA requirements intersect in very practical ways by using two common scenarios in the healthcare software ecosystem:

- Developing new software.
- Integrating new software modules or interfaces for a medical environment.

The expected audiences for this paper are business decision-makers, compliance managers, software developers, IT consultants, and systems integrators who are working within or on behalf of organizations that must meet HIPAA compliance requirements. This paper is not intended to advise organizations of their legal requirements and responsibilities. It is assumed that the reader understands the laws and regulations mentioned in this paper and how those laws and regulations apply to their organization. For readers unfamiliar with HIPAA, we provide a very brief overview in the **HIPAA Security Rule** section of this paper.

This paper can be used as a guide to speed adoption of the SDL to develop and integrate more secure software. It also is designed to demonstrate the areas where SDL practices and HIPAA requirements intersect.

SCENARIO 1: NEW SOFTWARE DEVELOPMENT

Developing new software represents the most straightforward application of the Security Development Lifecycle for a software provider who sells into a market with HIPAA requirements. For this scenario, we will use a fictitious company named "Contoso." Contoso represents a multinational conglomerate with a medical software division. This division is building a fully integrated Electronic Medical Record (EMR) product to address the needs of the mid-sized hospital market. For the purposes of this paper, this scenario will be referred to as the ***New Software Development Scenario***.

The software application in development will be used to perform the following functions in a hospital setting:

- Patient intake
    - Registration
    - Medical records
    - Abstracting (abstracting patient information for regulatory and insurance requirements)
    - Quality management and risk management
- Billing
- Capture, storage, and retrieval of patient charge information
    - Patient charge information
    - Patient demographic data
    - Bills, statements, claim forms, and logs
    - Receipts, adjustments, and refunds
    - Auto-proration
    - Account follow-up and management reports
- Records management
- Scanning and archiving
- Scheduling and referral management
- Patient care

    - Patient care and patient safety
    - Doctors' notes on patient presentation, diagnosis, and treatment, including follow-up
    - Electronic imaging
    - Laboratory results
    - Prescription tracking
    - Physician care management
    - Emergency department
    - Operating room management
    - Oncology management
- Accounting
    - Cost accounting
    - General accounting
- Human resource planning

Each of these functions will require access to and storage of patient data, generally classified as protected health information (PHI) by HIPAA. From the very early stages of software design, the software architects and developers of Contoso can apply application security practices through the SDL. By following these security best practices, architects and developers also help their organizations to meet some of the requirements of HIPAA and protect PHI at every step of the design and build process. Later in this paper we will go into the steps of the SDL process and how they apply to Contoso's effort to create its EMR application.

SCENARIO 2: MEDICAL INTEGRATION

In this scenario, a mid- to large-sized hospital has an EMR system installed and is engaged in the integration of several software modules from vendors other than the original EMR vendor. For the purposes of this paper, this scenario will be referred to as the ***Medical Integration Scenario*** These third-party modules include a(n)

- Picture archiving and communication system (PACS), which is used to store, deliver, and manipulate digital images, such as x-rays, magnetic resonance imaging (MRI), and nuclear medicine.

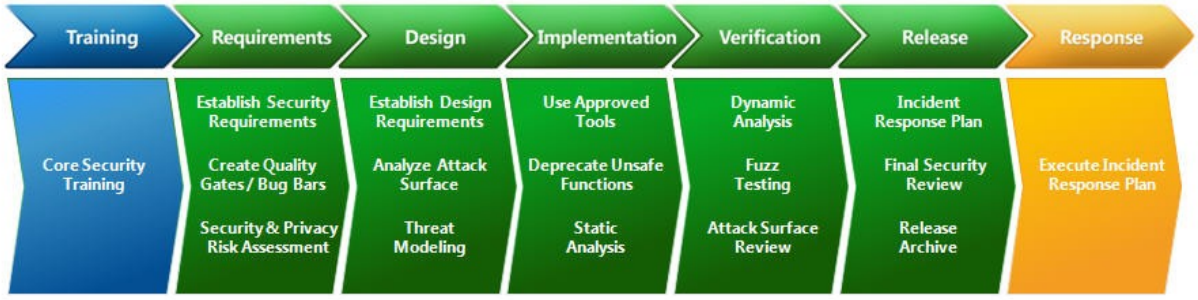
- Full-featured document management system for billing and records storage.
- Respiratory care suite with modules for both acute and chronic care.
- ePharmacy management module for dispensing, inventory, and reimbursement.
- Personal Health Record (PHR) system, the Microsoft HealthVault System.
- Insurance claims management module.
- Single sign-on access management system.

During the customization and integration of these modules, the SDL process can be used to ensure that the code developed to customize these modules is more secure. It can also provide a framework to evaluate the software security of the modules themselves prior to deployment. Even while performing integration, the disciplined security and privacy policies in the SDL complement and reinforce those HIPAA regulations designed to ensure that electronic PHI remains secure.

SECURITY DEVELOPMENT LIFECYCLE OVERVIEW

The Microsoft SDL is based on three core concepts—***education, continuous process improvement, and accountability.*** The ongoing education and training of technical job roles within a software development group is critical. The appropriate investment in knowledge transfer helps organizations to react appropriately to changes in technology and the threat landscape. Because security risk is not static, the SDL places heavy emphasis on understanding the cause and effect of security vulnerabilities and ***requires*** regular evaluation of SDL processes and introduction of changes in response to new technology advancements or new threats. Data is collected to assess training effectiveness, in-process metrics are used to confirm process compliance, and post-release metrics help guide future changes. Finally, the SDL requires the archival of all data necessary to service an application in a crisis. When paired with detailed security response and communication plans, an organization can use the SDL to provide concise and cogent guidance to all affected parties.

The SDL Process



The SDL process can be applied to any software development methodology, including Waterfall, Spiral, or Agile. It is also development-platform and operating-system agnostic. The security practices that comprise the SDL process are shown in Figure 1.

Figure 1: The SDL Process

Throughout this **SDL Process** section, this paper highlights the alignments between SDL practices and HIPAA Security Rules. This alignment is described in more detail in Appendix A of this paper.

The seven phases in the process are discussed individually, but it is important to notice that the five core phases roughly correspond to the following phases within the traditional ***software*** development life cycle:

- Training, policy, and organizational capabilities
- Requirements and design
- Implementation
- Verification, release, and response

It's essential to understand that the SDL is a security assurance process that is focused on integrating security into software development. Combining a holistic and practical approach, the SDL aims to reduce the number and severity of vulnerabilities in software. The SDL provides a continuous security and privacy perspective through all phases of the development process. Practical experience has shown that integrating security-specific activities within each part of a software development process is a cost-effective and measurable solution to security problems.

1.0 Pre-SDL: Security Training

1.1 Complete Core Security Training

All members of a software development team must receive appropriate training to stay informed about security basics and recent trends in security and privacy. Individuals in technical roles (developers, testers, and program managers) who are directly involved with the development of software programs must attend at least one unique security training class each year.

Basic software security training should cover foundational concepts such as:

- Secure design
    - Attack surface reduction
    - Defense in depth
    - Principle of least privilege
    - Secure defaults
- Threat modeling
    - Overview of threat modeling
    - Design implications of a threat model
    - Coding constraints based on a threat model
- Secure coding
    - Buffer overruns (for applications using C and C++)
    - Integer arithmetic errors (for applications using C and C++)
    - Cross-site scripting (for managed code and web applications)
    - SQL injection (for managed code and web applications)
    - Weak cryptography
- Security testing
    - Differences between security testing and functional testing
    - Risk assessment
    - Security testing methods
- Privacy
    - Types of privacy-sensitive data
    - Privacy design best practices
    - Risk assessment
    - Privacy development best practices
    - Privacy testing best practices

2.0 Requirements Practices

2.1 Establish Security Requirements

The need to consider security and privacy "up front" is a fundamental aspect of secure system development. The optimal point to define trustworthiness requirements for a software project is during the initial planning stages. This early definition of requirements allows development teams to identify key milestones and deliverables and permits the integration of security and

privacy in a way that minimizes any disruption to plans and schedules. Most, if not all, healthcare scenarios will benefit from these practices as illustrated through the two scenarios described in this paper.

Create a basic questionnaire to verify whether your product should be subject to the SDL. At a minimum, products that meet the following criteria should follow the SDL process:

- Any product that is commonly used or deployed within a business (for example, email or database servers).
- Any product that regularly stores, processes, or communicates personally identifiable information (PII) or PHI, such as financial, medical, or sensitive customer information.
- Any online products or services that target or are attractive to children.
- Any product that regularly touches or listens on the Internet.
- Any product that automatically downloads updates.

If the results of your questionnaire show that the SDL should be applied to your product, begin building your baseline security requirements from the content of the questionnaire.

Identify a security advisor who will serve as the first point of contact for security support and additional resources. This person will serve as the security advisor for the project. Identify the team or individual that is responsible for tracking and managing security for the product. This team or individual does not have sole responsibility for ensuring that a software release is secure, but this team or individual is responsible for coordinating and communicating the status of any security issues in the product. In smaller product groups, a single person may fill these roles.

Next, establish the minimum security requirements for the application as it is designed to run in its planned operational environment. Specify and deploy a security-vulnerability/work-item tracking system that will allow you to assign, sort, filter, and track completion of security-related bugs, work items, or tasks.

2.2 Create Quality Gates and Bug Bars

Quality gates and bug bars are used to establish minimum acceptable levels of security and privacy quality. Defining these criteria at the start of a project improves the understanding of risks associated with security issues and enables teams to identify and fix security bugs during development. A project team must negotiate quality gates (for example, all compiler warnings must be triaged and fixed prior to code check-in) for each development phase, and then have them approved by the security advisor, who may add project-specific clarifications and more stringent security requirements as appropriate. The project team must also demonstrate compliance with the negotiated quality gates in order to complete the Final Security Review.

A process should be defined to regulate the approval of exceptions to quality gates and bug bars throughout the lifecycle of your project. This exception process should require approval from both product team management and security experts who understand any potential risks associated with a security exception and can make plans for mitigation in both incident response planning and future product cycles.

2.3 Perform Security and Privacy Risk Assessment

Security risk assessments and privacy risk assessments are mandatory steps in the SDL that identify functional aspects of the software that require deep review. Such assessments must include the following information:

- (Security) Which portions of the project will require threat models before release?
- (Security) Which portions of the project will require security design reviews before release?
- (Security) Which portions of the project (if any) will require penetration testing by a mutually agreed upon group that is external to the project team?
- (Security) Are there additional testing or analysis requirements the security advisor deems necessary to mitigate security risks?
- (Security) What is the specific scope of the fuzz testing requirements?
- (Privacy) What is the privacy impact rating? The answer to this question is based on the following guidelines:
    - P1 High Privacy Risk: The feature, product, or service stores or transfers PII or PHI, changes settings or file type associations, or installs software.
    - P2 Moderate Privacy Risk: The sole behavior that affects privacy in the feature, product, or service is a one-time, user-initiated, anonymous data transfer (for example, the user clicks on a link and the software goes out to a web site).
    - P3 Low Privacy Risk: No behaviors exist within the feature, product, or service that affects privacy. No anonymous or personal data is transferred, no PII or PHI is stored on the machine, no settings are changed on the user's behalf, and no software is installed.

3.0 Design Practices

3.1 Establish Security Design Requirements

The design requirements activity contains a number of required actions. Examples include the creation of security and privacy design specifications, specification review, and specification of minimal cryptographic design requirements. Design specifications should describe security or privacy features that will be directly exposed to users, such as those that require user authentication to access specific data or user consent before use of a high-risk privacy feature. In addition, all design specifications should describe how to securely implement all functionality provided by a given feature or function. It’s a good practice to validate design specifications against the application’s functional specification. The functional specification should:

- Accurately and completely describe the intended use of a feature or function.
- Describe how to deploy the feature or function in a secure fashion.

Complete a security design review with a security advisor for any project or portion of a project that requires one. Some low-risk components might not require a detailed security design review.

To avoid costly mistakes, projects with a high privacy impact based on the Privacy Risk Assessment must hold a privacy design review.

Satisfy the minimal cryptographic design requirements established for your product when you established security requirements.

For additional details on this requirement, please read through the online SDL Process Guidance available at http://msdn.microsoft.com/en-us/security/cc420639.aspx.

3.2 Analyze Attack Surface

Attack surface reduction is a means of reducing risk by giving attackers less opportunity to exploit a potential weak spot or vulnerability. Attack surface reduction may include shutting off or restricting access to system services, applying the principle of least privilege, and employing layered defenses wherever possible. At a minimum, attack surface reduction should include the following:

- Use Code Access Security (CAS) correctly. When developing with managed code, use strong-named assemblies and request minimal permission. When using strong-named assemblies, do not use APTCA (Allow Partially Trusted Caller Attribute) unless the assembly was approved by a security review.
- Manage firewall exceptions carefully. Be logical and consistent when you make firewall exceptions. Any product or component that requires changes to the host firewall settings must adhere to the requirements that are outlined in Appendix D: Firewall Rules and Requirements, available at http://msdn.microsoft.com/en-us/library/cc307394.aspx.
- Ensure that your application runs correctly as a non-administrator.

By following these requirements, teams will reduce attack surface exposed by applications, increasing the security of the user and system.

3.3 Complete Threat Models

Threat modeling is used in environments where there is meaningful security risk. It is a practice that allows development teams to consider, document, and discuss the security implications of designs in the context of their planned operational environment and in a structured fashion. Threat modeling also allows consideration of security issues at the component or application level. Threat modeling is a team exercise, encompassing program/project managers, developers, and testers, and represents the primary security analysis task performed during the software design stage. Threat modeling activities include:

- Complete threat models for all functionality identified as high risk during the risk assessment practice from the Requirements phase. Threat models typically must consider the following areas:
    - All projects. All code exposed on the attack surface and all code written by or licensed from a third party.
    - New projects. All features and functionality.
    - Updated versions of existing projects. New features or functionality added in the updated version.

- Ensure that all threat models meet minimal threat model quality requirements. All threat models must contain data flow diagrams, assets, vulnerabilities, and mitigation. Threat modeling can be done in a variety of ways using either tools or documentation and specifications to define the approach. To learn more about the SDL Threat Modeling tool, visit http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx.
- Have all threat models and referenced mitigations reviewed and approved by at least one developer, one tester, and one program manager. Ask architects, developers, testers, program managers, and others who understand the software to contribute to threat models and to review them. Solicit broad input and reviews to ensure the threat models are as comprehensive as possible.
- Threat model data and associated documentation (functional and design specs) should be stored using the document control system used by the product team.

4.0 Implementation Practices

4.1 Use Approved Tools

All development teams should define and publish a list of approved tools and their associated security checks, such as compiler and linker options and warnings. This list should be approved by the security advisor for the project team. Generally speaking, development teams should strive to use the latest version of approved tools to take advantage of new security analysis functionality and protections.

4.2 Deprecate Unsafe Functions

Project teams should analyze all functions and application programming interfaces (APIs) that will be used in conjunction with a software development project, and prohibit those that are determined to be unsafe. Once this banned list is determined, project teams should use header files (such as banned.h and strsafe.h), newer compilers, or code scanning tools to check code (including legacy code where appropriate) for the presence of banned functions, and replace those banned functions with safer alternatives.

4.3 Perform Static Analysis

Project teams should perform static analysis of source code. Static analysis of source code provides a scalable capability for security code review and can help ensure that secure coding policies are being followed. The security team and security advisors should be aware of the strengths and weaknesses of static analysis tools and be prepared to augment static analysis tools with other tools or human review as appropriate.

5.0 Verification Practices

5.1 Perform Dynamic Code Analysis

Run-time verification of software programs is necessary to ensure that a program's functionality works as designed. This verification task should specify tools that monitor application behavior for memory corruption, user privilege issues, and other critical security problems. The SDL process uses run-time tools, along with other techniques such as fuzz testing, to achieve desired levels of security test coverage.

5.2 Perform Fuzz Testing

Fuzz testing is a specialized form of dynamic analysis used to induce program failure by deliberately introducing malformed or random data to an application. The fuzz testing strategy is derived from the intended use of the application and the functional and design specifications for the application. The security advisor may require additional fuzz tests or increases in the scope and duration of fuzz testing.

5.3 Conduct Attack Surface Review

It is common for an application to deviate significantly from the functional and design specifications created during the requirements and design phases of a software development project. Therefore, it is critical to re-review threat models and attack surface measurement of a given application when it is code complete. This review ensures that any design or implementation changes to the system have been accounted for, and that any new attack vectors created as a result of the changes have been reviewed and mitigated.

In addition, all security bugs identified in the project should be reviewed against the security bug bar and quality criteria established for your project to ensure that you have met the criteria or understand the potential attack surface associated with any bugs granted exceptions.

6.0 Release Practices

6.1 Create an Incident Response Plan

Every software release should include an incident response plan. Even programs with no known vulnerabilities at the time of release can be subject to new threats that emerge over time. The incident response plan should include:

- An identified sustained engineering team, or if the product team is too small to have these resources, an emergency response plan that identifies the appropriate engineering, marketing, communications, and management staff to act as points of first contact in a security emergency.
- On-call contacts with decision-making authority who are available 24 hours a day, seven days a week.
- Security servicing plans for code inherited from other groups within the organization.
- Security servicing plans for licensed third-party code, including file names, versions, source code, third-party contact information, and contractual permission to make changes (if appropriate).

6.2 Perform a Final Security Review

The Final Security Review (FSR) is a deliberate examination of all the security activities performed on a software application prior to release. The FSR is not a “penetrate and patch” exercise, nor is it a chance to perform security activities that were previously ignored or forgotten. The FSR usually includes an examination of threat models, exception requests, tool output, and performance against the previously determined quality gates or bug bars. The FSR results in one of three different outcomes:

- **Passed FSR.** All security and privacy issues identified by the FSR process are fixed or mitigated.
- **Passed FSR with exceptions.** All security and privacy issues identified by the FSR process are fixed or mitigated and/or all exceptions are satisfactorily resolved. Those issues that cannot be addressed (for example, vulnerabilities posed by legacy "design-level" issues) are logged and corrected in the next release.
- **FSR with escalation.** If a team does not meet all SDL requirements and the security advisor and the product team cannot reach an acceptable compromise, the security advisor cannot approve the project, and the project cannot be released. Teams must either address whatever SDL requirements that they can prior to launch or escalate to executive management for a decision.

6.3 Archive All Release Data

Software release must be conditional on completion of the SDL process. The security advisor assigned to the release must certify that the project team has satisfied security requirements. Similarly, for all products that have at least one component with a Privacy Impact Rating of P1, the project's privacy advisor must certify that the project team has satisfied the privacy requirements before the software can be shipped.

In addition, all pertinent information and data must be archived to allow for post-release servicing of the software. This includes all specifications, source code, binaries, private symbols, threat models, documentation, emergency response plans, license and servicing terms for any third-party software, and any other data necessary to perform post-release servicing tasks.

HIPAA SECURITY RULE

The Health Insurance Portability and Administrative Act (HIPAA) is a statute enacted by Congress, in part, to address longstanding problems in the healthcare insurance industry. Title II of HIPAA was directed toward administrative simplification and included requirements for the Department of Health and Human Services (DHHS) to develop standards to standardize, facilitate, and secure electronic transmission of health data.

A full discussion of the historical background and specific requirements of HIPAA are beyond the scope of this paper. Interested readers may consult the references in the appendix for further information or visit http://www.cms.hhs.gov/HIPAAGenInfo/.

The Security Rule was published in draft form in 2001. The final version of the Security Rule was released in 2003 and became effective in 2005. The Security Rule (codified at 45 CFR §§160-164) can be found on the web site of the U.S. Department of Health and Human Services, Office of Civil Rights at http://www.hhs.gov/ocr/privacy/.

Generally, the Security Rule establishes four overarching mandates for ***Covered Entities*** to:

- Ensure the confidentiality, integrity, and availability of all electronic PHI that the Covered Entity creates, receives, maintains, or transmits.

- Protect against any reasonably anticipated threats or hazards to the security or integrity of such information.
- Protect against any reasonably anticipated uses or disclosures of such information that are not permitted or required under HIPAA.
- Mandate workforce compliance with these requirements.

To accomplish these four overarching principles, the Security Rule lays out a risk analysis and risk management process to implement appropriate security requirements. Essentially, under this process, the Covered Entity must identify vulnerabilities (including their likelihood and impact, if exploited) and then develop a response, consistent with the Security Rule's standards, that brings such risks to a reasonable level.

Definitions

Before we begin our discussion of the Security Rule in detail, it is important to briefly review the definitions of some key terms used by the Security Rule:

- Protected Health Information
- Covered Entity
- Business Associate
- Standard
- Implementation Specifications
- Safeguards

Protected Health Information

Protected Health Information (PHI) is a subset of health information, in any media, including demographic information collected from an individual, that:

- Is created or received by a healthcare provider, health plan, employer, or health care clearinghouse;
- Relates to the past, present, or future physical or mental health or condition of an individual, the provision of health care to an individual, or the past, present, or future payment for the provision of health care to an individual;
- Identifies the individual or provides a reasonable basis to believe the information can be used to identify the individual; and
- Is not specifically excluded from the definition of PHI (generally, education, and employment records are excluded from HIPAA coverage). PHI may be in any media—oral, paper, electronic, etc.

Electronic PHI means PHI that is transmitted or maintained in electronic media. ***Electronic media*** means:

- Electronic storage media including memory devices in computers (hard drives) and any removable/transportable digital memory medium, such as magnetic tape or disc, optical disc, or digital memory card.

- Transmission media used to exchange information already in electronic storage media. ***Transmission media*** includes, by way of example, the Internet, extranet (using Internet technology to link a business with information accessible only to collaborating parties), leased lines, dial-up lines, virtual private networks, and the physical movement of removable/transportable electronic storage media. Certain transmissions, including transmissions of paper through facsimile and transmissions of voice through telephone are not considered to be transmission in electronic media. The rationale is that the information being exchanged did not exist in electronic form before the transmission.

Covered Entity

The Security Rule applies to what HIPAA calls a Covered Entity. In general, Covered Entities are:

- Health plans.
- Healthcare providers that engage in electronic HIPAA-covered transactions (for example, most hospitals, physicians, home health organizations, long-term care facilities).
- Healthcare clearinghouses.
- Sponsors of Medicare prescription drug cards.

Organizations and individuals who fall into one of these classes of Covered Entities must meet the requirements of the Security Rule with respect to their electronic PHI. This paper assumes the reader knows whether or not their organization is a Covered Entity for purposes of HIPAA applicability. Interested readers may learn more about the scope and applicability of the Security Rule in the references listed at http://www.hhs.gov/ocr/privacy/.

Business Associate

The healthcare ecosystem consists of more than just Covered Entities. Therefore, the Security Rule contemplates a category called Business Associates. A Business Associate is an organization or individual that performs, or assists with certain functions or activities on behalf of a Covered Entity that involves the use, creation, transmission, or storage of PHI. HIPAA recognizes the desirability that the safeguards and requirements that apply to Covered Entities also flow down to Business Associates. To address this common situation, HIPAA specifies that Covered Entities require their Business Associates to agree to put appropriate measures into place to safeguard PHI. This is typically done using contracts called Business Associate Contracts to govern the relationship between Covered Entities and Business Associates. Moreover, under the HITECH Act, Business Associates must comply with certain requirements of the Security Rule as if they were Covered Entities. Again, this paper assumes that the reader knows whether their organization is a Business Associate.

Standards

The use of the term ***standard*** can be confusing because it has numerous different uses in HIPAA, the Security Rule, and in every day usage, including:

- The HIPAA statue refers to Standard data elements or transactions, meaning that they comply with HIPAA-imposed requirements; for example, that something is a standard transaction.
- In the context of high-level HIPAA organization, Standard refers to the titles of the various regulations promulgated by DHHS; for example, Privacy Standards, Security Standards, or Transaction Standards. In this paper, we use the term Security Rule as opposed to Security Standards for clarity.
- The Security Rule defines Standard, in pertinent part as "a rule, condition, or requirement" describing classification of components, specification of materials, performance, or operations, or delineation of procedures for products, systems, services, or practices. It is used to refer to a group of related requirements that must be met by Covered Entities; for example, the Security Management Process Standard. In this use a Standard is often a higher level requirement or a goal and often has one or more detailed sub-requirements, which are called ***Implementation Specifications***.
- The colloquial term "Standard" used by engineering professionals generally is considered to be a set of specifications developed by committees of professionals, such as IEEE Std-802.3.

Implementation Specifications

Implementation Specifications are specific processes to reach the goals established by the Standards. They may be either REQUIRED or ADDRESSABLE. A REQUIRED Implementation Specification must be met by the Covered Entity. An ADDRESSABLE Implementation Specification is not optional in the usual sense; Covered Entities must evaluate the practicality of each ADDRESSABLE Implementation Specification in terms of the security risk and the implementation cost and feasibility, in light of their own situation. If reasonable under the circumstances, such Implementation Specification should be implemented as written. If deemed unreasonable, then an alternative approach should be implemented. The decision-making process must be documented.

Safeguards

Safeguards, which are not specifically defined in the Security Rule, are mechanisms, processes, or procedures used to mitigate security vulnerabilities and reduce security risks.

The Security Rule contains forty requirements for safeguards arranged in three general groups:

- Administrative Safeguards
- Physical Safeguards
- Technical Safeguards

APPLICABILITY OF THE SDL TO THE HIPAA SECURITY RULE

In this section we look at the phases of the SDL and discuss where the SDL specifically contributes to meeting HIPAA requirements. The practices we discuss in this section are mapped to the HIPAA requirements in a simple table included in Appendix A.

1.0 Pre-SDL: Security Training

In this Pre-SDL practice, the emphasis is on ensuring that all members of the development team have been trained in essential basic security and privacy requirements to give them the information they need to create more secure software. Since the SDL incorporates both security and privacy considerations, the basic tenets of the SDL training requirements are consistent with the HIPAA Security Rule, as well as the HIPAA Privacy Rule that require training of workforce, including the IT staff and developers as it relates to securing software to protect PHI.

2.0 Requirements Practices

Every product team must determine whether its solution is subject to the SDL based on a defined set of product security requirements. For example, a product team that wants to apply the SDL to a project developing line-of-business (LOB) or desktop software for a use by a Covered Entity where PHI is being accessed, stored, or transmitted must be assigned a security advisor and commit to the FSR.

A security advisor should track and manage the security and privacy issues discovered throughout the Security Development Lifecycle. Part of this management must include establishing security and privacy bug bars and an issue tracking system that allows dynamic tracking and closure of vulnerabilities.

The creation of these SDL requirements and issue-tracking processes allows a product team to verify resolution of any security issues that might adversely expose the customer or Covered Entity to additional unexpected security risks. Specifically, these processes would ensure that bugs that might result in a violation of HIPAA, because of a default software configuration setting, can be identified and addressed as early as the Design Phase of your Software Development Life Cycle (SDLC), potentially saving both development and compliance costs at the same time.

One example that shows how critical the identification of security requirements is to meeting HIPAA requirements is the New Software Development Scenario.

Using our earlier example, we have identified that PHI will be contained in the following modules of Contoso's EMR product:

- Patient intake
- Billing
- Records management
- Scanning and archiving
- Scheduling and referral management

- Patient care

Data Fields

A further examination of the data contained in the patient-intake module informs data security and privacy concerns with this data. The patient-intake module includes the data fields shown in Table 1.

Name	Address
Social Security Number (SSN)	Phone
Insurance Provider Name	Insurance Provider Group Number
Insurance Provider Member Number	Insurance Provider Plan Record
Next of Kin	Emergency contact
Emergency contact phone	Email address

Table 1: Patient-intake module data fields

The data in the patient-intake module used here is considered PHI because it can be used to establish a patient's identity and relates to the patient's condition, treatment, and payment for the health care services. Name, when linked with Address and SSN, can be used as a positive identity check. Additionally, the SSN, when combined with the Name, Insurance information, Phone, and Address, can be used to establish financial responsibility for healthcare. Likewise, access to Name, Address, SSN, and Phone can be used to positively identify someone's identity remotely (such as on the phone) and give an imposter access to healthcare and financial information.

Interfaces

EMR software must connect and exchange information with many other software systems because most systems cannot be completely self-contained. Security analysts have found that security and privacy are often at risk during the exchange of information between systems, and in the case of EMR systems, data must be both sent to and received by other systems.

Healthcare by its very nature includes contact with systems for both state and federal billing reimbursement, documentation archive and retrieval, federal and state reporting requirements, electronic data interchange with medical equipment, and transcript servers. For example, taking the Patient Intake- module being built by Contoso and applying it against an example of a mid-sized hospital, the EMR must exchange information with the systems shown in Table 2.

Interface	**Input/Output**	**Example System**
Institutional Submission 1	Output	Centers for Medicare and Medicaid Services (CMS) Billing
Institutional Submission 2	Output	CMS Billing
Professional Submission 1	Output	Professional Physician Billing through

Interface	Input/Output	Example System
		CMS
Professional Submission 2	Output	Professional Physician Billing through CMS
Archive Service	Output	Document Management Archive Service
OMNI BAR OV CHG to MT BAR	Input	Materials/supply vendor
OPTI BAR OV CHG to MT BAR	Input	Material Supply Vendor
Chart	Output	Electronic Charting Application
POC ISTAT	Input	Statistics reporting Application
State Submission	Output	State Billing System Medicaid
HCIS Mail	Output	EMR internal mail interface to email application
Medinet SORR	Input/Output	OR statistics
MM EDI	Input	EDI to specific medical equipment
PACS Interface	Input/Output	Interface to the x-ray system PACS (Picture Archiving and Communication System)
PHA DISP Suite	Input/Output	Pharmacy dispensing system
Transcript Suite	Output	Medical transcript system

Table 2: Example Input / Output to Electronic Medical Record Software

This example shows sixteen interfaces, each of which is a location for potential breach of security and privacy of PHI. In larger, more complex institutions, which may have regional and/or national scope, this number could easily triple. In both the New Software Development Scenario and the Medical Integration Scenarios, analyzing security and privacy during the integration of software should include applying threat modeling techniques and including threat mitigations in the requirements.

In this example, a privacy risk assessment was performed using the Initial Assessment Privacy Questionnaire found in the appendix of the SDL process documentation at http://www.microsoft.com/sdl. The Privacy Impact Rating measures the sensitivity of the data that the software will process from a privacy point of view and was discussed in the SDL Overview section. In both the New Software Development Scenario and the Medical Integration Scenario, a privacy impact rating of High Privacy Risk resulted from the privacy risk assessment.

A security bug bar using the STRIDE approach provides an excellent quality check for either scenario. The specific requirements of the HIPAA safeguards do not address the quality of the

software, but can relate to the overall category of security management process fairly easily. The security bug bar is more difficult to relate directly to HIPAA in the case of the Medical Integration Scenario than the New Software Development Scenario, because the bug bar only relates to the integration code in the Medical Integration Scenario whereas in the New Software Development Scenario it should be applied to all software developed.

3.0 Design Practices

Many of the Design Phase activities make it easier for Covered Entities to acquire, build, integrate, and maintain software in compliance with the HIPAA Security Rule. The best design practices that support the HIPAA Security Rule include the topics covered in this section.

3.1 Use of Appropriate Cryptographic Standards

The use of best practice cryptographic standards described here is the starting place to evaluate whether the software will help meet HIPAA Security Rule requirements. The SDL best practice recommendations are as follows:

- Use AES for symmetric encryption /decryption.
- Use 128-bit or better symmetric keys.
- Use RSA for asymmetric encryption /decryption and signatures.
- Use 2048-bit or better RSA keys.
- Use SHA-256 or better for hashing and message-authentication codes.

Prohibited and permitted actions are covered in the bulleted list that follows in Section 3.2 Attack Surface.

In the typical New Software Development Scenario, most users will require an open port in the firewall to perform their tasks. The code that listens on that firewall port must comply with certain quality requirements including cryptographic standards.

3.2 Attack Surface

There are a number of practices that enable least privilege in an application's default state or configuration and minimize the potential attack surface:

- All feature specifications should consider whether the feature should be enabled by default. Consider carefully whether to enable by default those features that are used infrequently. If a feature is not used frequently, then you should disable it.

- If the program needs to create new user accounts, then ensure that these accounts have as little permission as possible for the required function and that they also have strong passwords.

- Be very aware of access control issues. Always run code with the fewest possible permissions. When code fails, find out why it failed and fix the problem instead of increasing permissions. The more permissions any code has, the greater its exposure to attack.

- Default installation should be secure. Review functionality and exposed features that are enabled by default and constitute the attack surface carefully for vulnerabilities.

- Consider a defense-in-depth approach. The most exposed entry points should have multiple protection mechanisms to reduce the likelihood of exploitation of any security vulnerabilities that might exist. If possible, review public sources of information for known vulnerabilities in competitive products, analyze them, and adjust your product's design accordingly.

- If the program is a new release of an existing product, then examine past vulnerabilities in previous versions of the product and analyze their root causes. This analysis might uncover additional instances of the same classes of problems.

- Deprecate outdated functionality. If the product is a new release of an existing product, then evaluate support for older protocols, file formats, and standards, and strongly consider removing them in the new release. Older code written when security awareness was less prevalent almost always contains security vulnerabilities.

- Conduct a security review of all sample source code released with the product and use the same level of scrutiny as for the source code used to build object code released with the product.

- If the product is a new release of an existing product, then consider migration of any possible legacy code from unmanaged code to managed code. It is a basic best practice is to implement any new code as managed code whenever possible.

Attacks and threats evolve constantly, and staying current is important. You should utilize online resources and books to remain informed about security issues in the industry. Keep your development and integration team informed about new threats and vulnerabilities. Ensure that everyone on your team knows about unsafe functions and coding patterns. Maintain a list of your code's vulnerabilities; when you find new vulnerabilities, publish them in this list to your team.

3.3 Risk Analysis

Perform a risk analysis during the Design Phase of development by carefully reviewing security and privacy requirements and expectations to identify security concerns and privacy risks. It is efficient to identify and address these concerns and risks during the Design Phase.

As stated in the Interfaces section earlier, all interactions with other systems are potential points of data loss or breach of PHI and deserve special attention in the design considerations. Because almost all medical systems interface with other systems, interactions with other systems become a major design point and any interfaces must be secure by design.

External code (source or object)

Any part of the system using source or object code obtained from an external source should be evaluated from a security design perspective.

In our Medical Integration Scenario, the core EMR is based on .NET and has been designed using the SDL process. If, however, we look at each of the modules slated for integration, we find each is slightly different with respect to source code and object code. In short, they come from a large variety of sources that approach security in a variety of ways. In our scenario, we need to analyze the risk of each module by using the SDL as a framework. This sort of risk analysis could result in a risk analysis table similar to what is represented in Table 3.

Type of Software	Approach to Security	Likely Risk
PACS	The PACS is based on Java and was designed to hold PHI using a rigorous security model similar to SDL. It comes from one of the largest medical software and imaging companies in the world and is expected to be top notch from a security and privacy perspective.	Low
Document Management System	The document management solution chosen is an open source solution that uses the Python language for integration. This solution was written with minimal security considerations and must be carefully examined for the ability to provide security to the large amount of PHI included in it.	High
Respiratory Care Suite	A respiratory care suite with modules for both acute and chronic care has been selected for the company in the Medical Integration Scenario. This software was subcontracted by the respiratory care company and is of unknown security design. To integrate it into the overall EMR, a design review and manual code review of the software will need to be done, and an extensive question and answer session with the most knowledgeable personnel at the respiratory care company.	High
ePharmacy Management Module	An ePharmacy management module for dispensing, inventory, and reimbursement has been selected for the Medical Integration Scenario. This module is the leading ePharmacy module and has been developed using SDL-like security practices that have been verified by the security team.	Low
Personal Health Record System	The Microsoft HealthVault System PHR was selected by the company in the Medical Integration Scenario.	Low
Claims Management Module	A claims management module, which was originally built by a company that now has been purchased by one of the largest medical insurance providers in the U.S., has been selected. The original software was built on .NET, but has a large amount of custom code, including machine code to deal with the EDI requirements of state	High

Table 3: Example Risk Analysis on Integration Scenario

This risk analysis practice provides an organization with a high-level approach for evaluating and understanding how each integrated module affects the security of its PHI. Further details of the integration security risk can be quite extensive and are beyond the scope of this paper.

If your project has a Privacy Impact Rating of High, then identify a compliant design based on the concepts, scenarios, and rules in the Microsoft Privacy Guidelines for Developing Software Products and Services found in Appendix C of the SDL Process documentation. Appendix C: SDL Privacy Questionnaire at http://msdn.microsoft.com/en-us/library/cc307393.aspx is available.

3.4 Threat Modeling

For security concerns, ***threat modeling*** is a systematic process that is used to identify threats and vulnerabilities in software. You must complete threat modeling during project design. A team cannot build secure software unless it understands the assets the project is trying to protect, the threats and potential vulnerabilities introduced by the project, and details of how the project mitigates those threats. Microsoft has created the SDL Threat Modeling Tool to help with threat modeling. It can be found at http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx.

Threat models should be completed for all functionality identified during the Requirements Phase. Threat models typically must consider the following areas:

- **All projects.** All code exposed on the attack surface and all code written by or licensed from a third party. This applies to all interfaces in the case of EMR into which third-party modules are integrated, as is the case in the Medical Integration Scenario.
- **New projects.** All features and functionality. This applies in the New Software Development Scenario.
- **Updated versions of existing projects.** New features or functionality added in the updated version. This applies to third-party modules added to a purchased EMR, as in the Medical Integration Scenario.

4.0 Implementation Practices

In the implementation phase, recommended portions of the SDL process make it easier for Covered Entities to be HIPAA compliant.

- Implementation of user-facing security documentation maps to the HIPAA Security Rule requirement for education and training on security, as does the recommended practice of making information about secure configurations part of the default product documentation.
- The implementation of the best practice of attack surface reduction recommended in the SDL does not have a direct call out in the HIPAA Security Rule, but relates to part of the HIPAA requirement for information access management, as well as the security management process, involving risk analysis and risk management.
- The best practice of informing users about security best practices (such as removing guest accounts and default passwords) maps to the requirements of the risk analysis, risk management, information access, and password management sections of the HIPAA Security Rule.

- The SDL best practice of describing all ports and communication channels maps to the requirement of access management, access authorization, and information system activity monitoring requirements in the HIPAA Security Rule.
- The SDL best practice of creating privacy deployment guides for customers, ties into requirements of the risk analysis, risk management, information access, and password management requirements of the HIPAA Security Rule. This also is consistent with the HIPAA Privacy Rule, which is beyond the scope of this paper.
- The SDL best practice of documenting privacy best practices for the development team maps to the education and training section of the HIPAA Security Rule. Again, this also is consistent with the HIPAA Privacy Rule, which is beyond the scope of this paper.
- Examples of the available software security tools include Microsoft Anti-Cross Site Scripting Library V3.1, FxCop, and CAT.NET. While useful in the New Software Development Scenario, these tools are particularly useful for web-based applications and should be used by any organization attempting to integrate a web-based system similar to the Medical Integration Scenario. The use of tools is not specifically called out in the HIPAA Security Rule; however, the tools would meet some of the requirements of the security management process standard.

The implementation section of the SDL process is critical to the overall ability of the healthcare software to protect the patient PHI, particularly in the case of the New Software Development Scenario. In writing new software, the introduction of security early in the lifecycle - especially within the Implementation Phase - is a core requirement for writing more secure software.

In the Medical Integration Scenario, where the software is written by a third party, vendors should be contractually obligated to practice SDL implementation. Security tools are not always 100 percent effective at catching potential security issues at the point of integration. Using good security tools, validating that unsafe functions are not used, and performing static analysis on the source/object code proactively will reduce attack surface and improve the security and integrity of a solution.

5.0 Verification Practices

The Verification Phase of the SDL verifies that the security best practices used in the preceding Requirement, Design, and implementation Phases actually have worked as intended. It also provides a quality measurement tool in the bug bar, extensive testing using several methods, and an automated or manual code review for security issues. In this section, we identify several SDL verification phase best practices that help a Covered Entity meet HIPAA Security Rule requirements.

- A best practice of the Verification phase is the proactive security testing of the software under development using fuzz testing. In fuzzing, random instructions are sent to the software being tested to ensure that security is not breached. It is particularly useful in any web-based applications and should be used by the development organization in the Medical Integration Scenario. Fuzzing is a good way to test the modules planned for integration and integration code of the modules in the Medical Integration Scenario. If the module is responsible for document or file transaction, then a specific type of fuzz

testing, such as file fuzzing, is required. Since all of the SDL best practices cannot be implemented on existing software you may have acquired from a third-party company, the use of fuzzing increases the confidence that PHI in these modules is safe and secure.

- The SDL practice of secure code review in the Verification Phase provides quality assurance in both the New Software Development Scenario and the Medical Integration Scenario. The amount of code will differ between the two scenarios, and the depth of the review will be much deeper in the case of the New Software Development Scenario. This best practice, while not specifically called out in the HIPAA Security Rule, will meet some of the items in the high-level requirements of the security management process.

Much of the Verification phase of the SDL process is particularly critical in the pre-deployment activities associated with the Medical Integration Scenario. Since the modules already are built, the Requirement, Design, and Implementation Phases are past. The best practices of the Verification Phase can use testing techniques to verify the software security of these modules and provide a higher assurance of security than analysis alone.

6.0 Release Practices

The Release Phase of the SDL process is where an incident response plan is created and a final security and privacy review is completed. It also is where all documentation on the software is finalized and archived for reference in security response and next-version/future product development.

- The SDL best practice of performing a privacy review prior to release and addressing all privacy bugs prior to release is useful in both the New Software Development Scenario and the Medical Integration Scenario. The privacy review is not specifically mandated in the HIPAA Security Rule, but it is a logical part of the security management process.
- The SDL best practice of performing a Final Security Review (FSR) for security bugs can be useful in both scenarios. A FSR is not specifically named in the HIPAA Security Rule, but it does contribute toward fulfilling the needs involved with the high-level categories of the security management process and information management access.

The formal Release Phase of any software project is very important to the customers of the software because a formal release improves the quality of the software tenfold. Release is where the final quality checks are performed, all items are inspected, and the true professionalism of the software development organization shows through.

6.1 Response Plan

In the response plan, the organization defines their strategy to react to security incidents that may occur after release or deployment. The only difference in response between the cases of the New Software Development Scenario and the Medical Integration Scenario is the size and complexity of the response plan. In the case of the New Software Development Scenario, the response will be much more complex because the bulk of the response must come from the development organization. In the case of the Medical Integration Scenario, the organization that has integrated the solution should create a response plan as well as ensure the providers of the modules have also created a response plan that will provide the sustained engineering and response efforts. However, the organization using the module still will need to coordinate and ensure a quality response.

The SDL best practice of response planning is extremely critical to the safety and security of the New Software Development Scenario and the Medical Integration Scenario. It also specifically addresses the requirements for security incident procedures and contingency plans called out as requirements in the HIPAA Security Rule. The response plan provides management with a way to think through what the response will be when the inevitable security vulnerability is found in the software, and have a plan of record as to how to engage with the development community to fix the vulnerabilities. This response plan must address both security and privacy, and include the contacts if a problem occurs and a method of engaging with those contacts.

THE CHALLENGE OF HIPAA

HIPAA creates challenges for Covered Entities, their Business Associates, and software developers when they are faced with validating their application security. To address some of these challenges, such organizations can use the Microsoft SDL to validate that the product development team has performed the practices necessary to improve the security of the product. For developers, the SDL provides actionable practices for implementing more secure software.

Covered Entities face enforcement risk from the federal government with respect to compliance with HIPAA and the HITECH Act. This enforcement extends to the various states in which they do business (and potentially from where they draw their patients) with respect to each state's confidentiality laws, most of which are not preempted by HIPAA. Additionally, the HITECH Act created the first Federal breach notification law. This obligation is coupled with the breach notification laws in the majority of the states, which in general have been based loosely on California's law.

Since the passage of the HITECH Act, DHHS, as well as state Attorneys General, appear to be increasing HIPAA enforcement activity.

HIPAA is heavily focused on the management of risk. There have been recent changes in the management of IT risk with the release of the Information Security Forum (ISF) Risk Assessment Methodology and the RiskIT Framework from NIST and ISACA.org, which provides a systematic way to identify, codify, and manage IT-related risk. Covered Entities can use structured and systematic methods like these to demonstrate how they are proactively addressing security risk. ISACA is an international body with the backing of worldwide accounting and auditing standards bodies.

The use of a risk management standard promotes an organization's response to the HIPAA Security Rule that will evolve over time to address new risks as they arise. This ongoing nature of the standard ensures the security and privacy of PHI and allows the approaches to risk mitigation to grow as technology changes. The SDL fits well with that approach because it includes a baseline set of security best practices to mitigate potential risks in software development while also providing a framework for continuous improvement to the methodology to meet the ever-changing risks emerging in the healthcare software ecosystem.

CONCLUSION

In this paper, we have looked at how the Security Development Lifecycle (SDL) helps an organization meet some of the requirements of the HIPAA Security Rule. We have provided an overview of how a software developer can use the SDL process to meet many of the

requirements of the HIPAA Security Rule while also creating or integrating more secure software. Throughout the paper, we have tried to associate these activities with common healthcare software scenarios where the HIPAA Security Rule would be a consideration, and how the SDL process might help structure the protection of PHI in healthcare software.

Our hope is that by reading this paper, an organization that is writing or integrating software in a HIPAA regulated environment can readily see how the security best practices of the SDL can help a Covered Entity meet many of the HIPAA Security Rule requirements. We have shown that adding SDL practices to a software development process provides a methodology for securing software during development that can be applied to some HIPAA-regulated scenarios. We hope this paper will be a valuable resource in applying the SDL to software development and integrated software modules in healthcare environments.

FURTHER READING

For further reading on the both HIPAA and the SDL, the following publications are recommended:

- A Guide to HIPAA Security and the Law, ed. Steve Wu, Am Bar Assoc 2007
- The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software, Steve Lipner and Michael Howard, MS Press, 2006.
- Simplified Implementation of the Microsoft SDL (http://go.microsoft.com/?linkid=9708425)
- The SDL Optimization Model (http://www.microsoft.com/security/sdl/getstarted/assess.aspx)
- The Microsoft Security Development Lifecycle process guidance – Version 5 (http://go.microsoft.com/?linkid=9724944).

REFERENCES

1. Protected Health Information in paper and verbal form is covered by the Privacy Rule, which can be found at http://www.hhs.gov/ocr/privacy/.
2. HIPAA Privacy and Security by Francoise Gilbert, in Stephen Wu, ed. A Guide to HIPAA Security and the Law, American Bar Association, 2007
3. 45 CFR §160,
4. 45 CFR §164 (subparts A, C, D, and E)
5. 45 CFR §162 (subparts A, D, F, I-R)
6. http://www.ieee.org/web/standards/home/index.html

APPENDIX A

Administrative, Physical, and Technical Safeguards mapped to the SDL practice areas.

NOTE: Blank sections of the below table indicate areas where there is no clear mapping between Administrative Safeguards and SDL Practices.

Table 1. Administrative Safeguards: 45 CFR §164.308

Requirement		45 CFR	Text	SDL Practice	Additional SDL Notes	
Security Management Process (STANDARD)		§164.308(a)(1)	Implement policies and procedures to prevent, detect, contain, and correct security violations.			
	Risk Analysis (REQUIRED)	§164.308(a)(1)(ii)(A)	Conduct an accurate and thorough assessment of the potential risks and vulnerabilities to the confidentiality, integrity, and availability of electronic protected health information held by the covered entity.		2.1 Security Requirements 2.3 Security and Privacy Risk Assessment 3.3 Threat Modeling Informed by: 6.2 Final Security Review	The risk analysis information created by the Risk Assessment and Threat Modeling activities of the SDL should contribute to the overall organizational risk analysis that a customer must undertake to comply with this implementation specification.
	Risk Management (REQUIRED)	§164.308(a)(1)(ii)(B)	Implement security measures sufficient to reduce risks and vulnerabilities to a reasonable and appropriate level to comply with 45 CFR §106.306(a).		2.1 Security Requirements 2.2 Quality Gates and Bug Bars 2.3 Security and Privacy Risk Assessment 3.1 Design Requirements 3.2 Attack Surface Reduction 3.3 Threat Modeling	The SDL process itself focuses on the development of software products and services while minimizing security vulnerabilities. In SDL 5.0 -Line-of-Business (LOB) p. 75, compliance reviews are required, providing a means for assessing whether a given implementation meets the security and privacy requirements. SDL 5.0 Integration-Points Design Review (p. 20) SDL 5.0 Hardware Review
	Sanction Policy (REQUIRED)	§164.308(a)(1)(ii)(C)	Apply appropriate sanctions against workforce members who fail to comply with the security policies and procedures of the covered entity.		2.1 Security Requirements 3.1 Design Requirements	While the SDL does not directly address the implementation of the sanction policy, the SDL does require LOB development teams to consider product requirements, such as user account deactivation or de-provisioning (p. 64) that may become part of the sanction policy implementation (removing or limiting a user's access rights promptly). In addition, the SDL requires

Table 2. Physical Safeguards: 45 CFR §164.310

Requirement		45 CFR Section	Text	SDL Practice	SDL Notes	
Facility Access Controls (STANDARD)		§164.310(a)(1)	Implement policies and procedures to limit physical access to its electronic information systems and the facility or facilities in which they are housed, while ensuring that properly authorized access is allowed.			
	Contingency Operations (ADDRESSABLE)	§164.310(a)(2)(i)	Establish (and implement as needed) procedures that allow facility access in support of restoration of lost data under the disaster recovery plan and emergency mode operations plan in the event of an emergency.			
	Facility Security Plan (ADDRESSABLE)	§164.310(a)(2)(ii)	Implement policies and procedures to safeguard the facility and the equipment therein from unauthorized physical access, tampering, and theft.			
	Access Control and Validation Procedures (ADDRESSABLE)	§164.310(a)(2)(iii)	Implement procedures to control and validate a person's access to facilities based on their role or function, including visitor control, and control of access to software programs for testing and revision.			
	Maintenance Records (ADDRESSABLE)	§164.310(a)(2)(iv)	Implement policies and procedures to document repairs and modifications to the physical components of a facility which are related to security (for example, hardware, walls, doors, and locks).			
Workstation Use (STANDARD)		§164.310(b)	Implement policies and procedures that specify the proper functions to be performed, the manner in which those functions are to be performed, and the physical attributes of the surroundings of a specific workstation or class of workstation that can access electronic protected health information.			
Workstation Security (STANDARD)		§164.310(c)	Implement physical safeguards for all workstations that access electronic protected health information, to restrict access to authorized users.			
Device and Media Controls (STANDARD)		§164.310(d)(1)	Implement policies and procedures that govern the receipt and removal of hardware and electronic media that contain electronic protected health			

Table 3. Technical Safeguards: 45 CFR §164.312

Requirement	45 CFR	Text	SDL Practice	SDL Notes
Access Control (STANDARD)	§164.312(a)(1)	Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights as specified in §164.308(a)(4).	2.1 Security Requirements 3.1 Design Requirements 3.2 Attack Surface Reduction	SDL 5.0 Design Phase (pp. 17-24)
Unique User ID (REQUIRED)	§164.312(a)(2)(i)	Assign a unique name and or number for identifying and tracking user identity.	2.1 Security Requirements 2.3 Security and Privacy Risk Assessment 3.1 Design Requirements	SDL 5.0 Design Phase (pp. 17-24)
Emergency Access Procedure (REQUIRED)	§164.312(a)(2)(ii)	Establish (and implement as needed) procedures for obtaining necessary electronic protected health information during an emergency.	3.1 Design Requirements	SDL 5.0 Design Phase (pp. 17-24)
Automatic Log-off (ADDRESSABLE)	§164.312(a)(2)(iii)	Implement electronic procedures that terminate an electronic session after a predetermined time of inactivity.	2.1 Security Requirements 3.1 Design Requirements	SDL 5.0 Web Application Logout (p. 21)
Encryption and decryption (ADDRESSABLE)	§164.312(a)(2)(iv)	Implement a mechanism to encrypt and decrypt electronic protected health information.	2.1 Security Requirements 3.1 Design Requirements	SDL 5.0 Discussion of Crypto (p. 18)
Audit Controls (STANDARD)	§164.312(b)	Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information.	2.1 Security Requirements 3.1 Design Requirements	SDL 5.0 LOB Enabled for Auditing (p. 21)
Integrity (STANDARD)	§164.312(c)(1)	Implement policies and procedures to protect electronic protected health information from improper alteration or destruction.	2.1 Security Requirements 3.1 Design Requirements	SDL 5.0 Design Phase (pp. 17-24)
Mechanism to authenticate electronic protected health information	§164.312(c)(2)	Implement electronic mechanisms to corroborate that electronic protected health information has not been altered or destroyed in an unauthorized manner.	2.1 Security Requirements 3.1 Design Requirements	SDL 5.0 Design Phase (pp. 17-24)

Changes Made:

- 6/28/10: Page 32 - Emergency Mode Operations Plan ~~(ADDRESSABLE)~~ corrected to (REQUIRED)
- 6/28/10: Corrected multiple typos