Interested in learning more about security?

SANS Institute InfoSec Reading Room

A Context-Based Access Control Model for HIPAA Privacy and Security Compliance

It is somewhat ironic that the Administrative Simplification subtitle of HIPAA, the Health Insurance Portability and Accountability Act of 1996, includes several provisions that result in administrative complexity of a hitherto unknown degree. Cumbersome record keeping and reporting requirements offset the benefits brought about by claims-processing automation. This paper proposes a new approach to meeting much of the burden imposed by the HIPAA privacy and security requirements. By adopting this approach, an organizat...

Submitted by: Harry E. Smith
Course: SANS Security Essentials (GSEC)
Version: v1.2e

A Context-Based Access Control Model for HIPAA Privacy and Security Compliance

Harry E. Smith, CISSP
July 18, 2001

It is somewhat ironic that the Administrative Simplification[1] subtitle of HIPAA, the Health Insurance Portability and Accountability Act of 1996, includes several provisions that result in administrative complexity of a hitherto unknown degree. Cumbersome record keeping and reporting requirements offset the benefits brought about by claims-processing automation. This paper proposes a new approach to meeting much of the burden imposed by the HIPAA privacy and security requirements. By adopting this approach, an organization may avail itself of the significant benefits brought about by transaction standardization and, at the same time, avoid the unwanted side effects of having to adopt inconvenient manual processes.

Wherefore HIPAA Privacy and Security?[2]

In the late 1980s it was becoming increasingly apparent that health care costs were skyrocketing. It was clear to most analysts that a large component of the cost of health care was the need of every health care provider to have people on staff dedicated to dealing with the mountains of paperwork involved in processing health insurance claims. Clearly, it would be to everyone's advantage to streamline the claims processing system.

The first attempt to do so occurred in 1991 when Dr. Louis Sullivan, then Secretary of the Department of Health and Human Services (HHS) in the first Bush administration, proposed a "single-payer" scheme to deal with the waste involved in using hundreds of different claims forms, each requiring special handling. To the insurance companies, this sounded too much like an attempt to nationalize their industry. In response, they formed WEDI, the Workgroup on Electronic Data Interchange, attempting to demonstrate that the private sector could deal with the problem. The WEDI approach was to adopt a set of standards for electronic claims processing transactions that would all but eliminate paperwork and adjudicate health care claims in a matter of days rather than months.

WEDI ran into trouble because of a lack of a central enforcement mechanism. Individual insurance companies agreed to use a core set of common transaction formats, but reserved the right to employ non-standard code sets. The result was the replacement of a multiplicity of paper forms with a multiplicity of proprietary transaction codes. Needless to say, software developers did not stampede to offer commercial products to implement these new "standards."

The Clinton Health Care Reform Initiative of 1993 tried to impose some discipline on the WEDI effort. Unfortunately it imposed a great deal of discipline in several other areas as well. Ira Magaziner produced a 600 page rulebook that smacked of "big government" trying to control things that many believed should be left to free market mechanisms. So the initiative failed to attract popular support and was abandoned.

In 1996 Senators Edward Kennedy and Nancy Kassebaum introduced a bill to deal with the problem of people losing their health care coverage when they changed jobs. To counter criticism that providing health insurance portability would result in excessive premium costs, the bill's supporters added the "Administrative Simplification" subtitle. Administrative Simplification promised to enforce transaction and code set standards and to deliver the benefits promised by the WEDI initiative.

At the same time that the full benefits of automated claims processing were coming into view, people began to be concerned about the problem of privacy. The unauthorized disclosure of Arthur Ashe's positive HIV test was in the news. "If these kinds of privacy violations are possible using our current manual systems," members of Congress asked, "what is going to happen when all of our medical records are stored on computers and transmitted over the Internet?" The result of this concern was that the HHS regulations promulgated in support of HIPAA would include privacy and security standards in addition to transaction and code set standards.

The privacy regulation has to do with the rights of individuals who are the subjects of health information that could prove embarrassing or damaging if improperly disclosed (I use the term "subject individuals" to refer to people who are the subjects of health information that is used or maintained by an organization.) The security regulation deals with the storage, processing and transmission of medical data. In a sense, security is the "vehicle" and privacy is the "payload."

Privacy and Security Principles

The HIPAA privacy regulation is based on seven "safe harbor privacy principles" articulated in an agreement negotiated between the United States Department of Commerce and the European Union in July of 2000.[3] These principles are:

- **Notice** – The subject individual has a right to know what information is maintained and how that information is used or disclosed.

- **Choice** – The subject individual has the right to control the uses and disclosures of sensitive information.

- **Onward Transfer** – The subject individual has a right to control subsequent uses and disclosures of sensitive information.

- **Security** – Those who store, process or transmit sensitive information have an obligation to protect its confidentiality.

- **Data Integrity** – Those who store, process or transmit sensitive information have an obligation to prevent unauthorized alterations.

- **Access** – The subject individual has a right to inspect sensitive information to ensure its accuracy and completeness.

- **Enforcement** – The subject individual has a right to redress of privacy violations.

The HIPAA security regulation is based on the principles set out in For the Record[4], a publication of the National Academy of Sciences. This 1997 document is the result of extensive study by the "Committee on Maintaining Privacy and Security in Health Care Applications of the National Information Infrastructure." In the words of the report, the committee sought to address:

- "**Threats to health care information** - What problems have health care organizations encountered to date regarding unauthorized access to individually identified patient data? To what extent has the security of health information systems been compromised or threatened by the introduction of electronic medical records and networked information systems? What problems could be encountered in the future related to unauthorized access to individually identifiable patient data? How significant is the threat posed by inferential identification through the linking of databases with unidentifiable information?

- **Adequacy of existing privacy and security measures** - What types of policies are in place to provide privacy, security, and confidentiality? How adequate are these policies in practice? What technical features are incorporated into health information systems to provide security? How effective are they? What has been done to educate users about the need for privacy and security and their responsibilities for protecting health information?

- **Future mechanisms and best practices** - What other approaches to information privacy and security are worthy of testing in health care organizations? What approaches should be broadly promulgated? How cost-effective are various approaches? What combination of technologies, policies, and standards would help to promote better information security for health-related data? How can highly sensitive aspects of an individual's health care records (e.g., mental health history and HIV status) be better protected?

- **Barriers to adoption** - What barriers exist to the adoption of better information security practices and technology (e.g., cost, ease of use)? What incentives are needed to encourage providers to adopt sound information privacy and security practices and to secure health information systems?"

The result of this analysis is a comprehensive document that addresses the protection of health information from technical, administrative and procedural vantage points.

The Problem To Be Solved

Just about everyone agrees that the cost savings that will accrue as a result of uniform adherence to the transaction and code set standards will be substantial. Some HIPAA enthusiasts believe that they amount to what Andrew Grove[5] calls a "strategic inflection point," that will fundamentally alter the health care system in ways that are impossible to predict at this time. Unfortunately, some provisions of the HIPAA regulations threaten to take us backward into an increasing amount of paper processing and manual intervention.

The most apparent "paper" requirement is the need for signed consent and authorization forms. Health care providers who have a direct treatment relationship with their patients must obtain a signed consent that allows them to use or disclose protected health information (PHI) for purposes of treatment, payment or health care operations. All "covered entities" (i.e. health plans, health care clearinghouses and health care providers who transmit PHI in electronic form) must obtain a signed authorization to use or disclose PHI for other purposes such as marketing or fundraising.

HHS has published a proposed electronic signature standard in the Federal Register that will meet the consent and authorization requirements once it is finalized. The problem is that there is no ubiquitous PKI infrastructure with which the vast majority of the public is familiar. So consent and authorization to use or disclose PHI will likely be implemented on paper for some time to come.

The crux of the problem lies in the manual checking that will have to be done and in the mandatory record keeping associated with the use and disclosure of PHI. Obtaining a signed consent will probably not be an extraordinary burden on physicians; they are already required to obtain the patient's consent before providing treatment, and the two consent forms may be combined in a single document. But an exception to the consent rule complicates the process.

According to § 164.506(a)(1)[6], a signed consent form allows a health care provider to use or disclose PHI according to his published "notice of information practices." The exception is found in § 164.502(c), whereby the patient may request a restriction on one or more practices described in the notice. If the physician agrees to the restriction, then he or she is bound by it. This means that, prior to each use or disclosure, the physician must be mindful of any restrictions that are in effect. If this involves even a small amount of manual effort to check the patient's file, the compound impact over many encounters can be significant.

Keeping track of consent forms and restriction requests may not seem like a prohibitively costly complication, but this is merely the tip of the iceberg. For almost every HIPAA privacy or security provision there is an exception. Occasionally there is an exception to the exception. And once in a while there is an exception to the exception to the exception.

Consider, for example, the right of a patient to inspect and/or obtain a copy of his or her own medical records. This is provided for in § 164.502(a)(2)(i); a covered entity must grant access to

PHI to the subject individual when requested. But by § 164.524(a)(1), the covered entity is not required to release certain kinds of information such as psychotherapy notes (an exception). At the same time § 160.203(b), would require granting access to this type of information in accordance with a more stringent state law if such a law exists (an exception to the exception). But on the other hand, one must bear in mind § 164.528(a)(2)(i), which requires that access to protected health information by the subject individual be suspended temporarily in response to requests by law enforcement or public health officials (an exception to the exception to the exception). This is only one such example.

If these and similar issues must be resolved manually, by a staff member opening a filing cabinet, individually scanning the contents of each patient's folder and reflecting deeply on all of the permutations of individual regulatory provisions, then the golden promise of automated claims processing will soon turn to dross. On the other hand, if systems can be designed that automate the myriad of complex access control decisions that must be made to properly implement the HIPAA privacy and security standards, then the undesirable side effects can be eliminated or at least drastically reduced. The access control model that is presented here is designed to accomplish this automation.

Databases and Database Security

Some readers may remember IBM's introduction of the Information Management System (IMS) in the mid 1970s. IMS was the first example of what came to be known as "database" technology. Prior to the arrival of IMS, information was kept in individual sequential files. If an application wrote to several output files, then it was necessary to ensure that whenever one of the files was restored from a backup tape, that the related files would be restored as well to prevent an "out of sync" condition. Database technology solved this problem by keeping track of related data repositories and maintaining the integrity of the entire structure. This "coordinated recovery and restart" capability was the single most important contribution of database technology. The contingency planning requirements set out in §142.308(a)(3) of the HIPAA security standard call for exactly this type of functionality.

The reason why IMS is not the database of choice today is that it (and similar database products based on a "hierarchical" model) required the acquisition and use of a lot more skill than suited the average programmer. To use IMS efficiently one had to attend many hours of training, read many pages of technical manuals and steal as much code as possible from the "gurus." Applications that used more than the bare essential database features were extremely difficult to develop and maintain. The few surviving IMS programmers today are worth their weight in gold to hundreds of organizations running arcane applications written by programmers who have long since retired or gone insane.

Thankfully, E. F. Codd, in a series of academic papers[7] published throughout the 1970s, gave us the "relational" database model. Relational databases stored data in tabular form, where each row contained a single database record and each column represented a record field. Now a programmer could retrieve information from the database using structured query language (SQL[8]) that was as simple as

```
SELECT <one or more items>
FROM <one or more tables>
WHERE <one or more conditions are met>
```

Furthermore, the coordinated restart and recovery capabilities were improved in such relational database implementations as Oracle's Database Server and IBM's DB2.

Basic relational database security is achieved by the use of GRANT and REVOKE commands. When an individual user is granted SELECT, INSERT, UPDATE or DELETE privileges on a group of relational database tables, this process is referred to as "user-based access control." For a large number of users who need selective access to a large number of tables the user-based access control scheme becomes quite cumbersome. For this reason, most modern relational database products include the concept of "roles."

Roles are access privilege groups in the database world. A distinct role can be defined for each group of users who have common access requirements. Typically this categorization is based on assigned job responsibilities. Table access privileges are assigned to roles and roles are, in turn, granted to users. This greatly simplifies the privilege management task of the database administrator. This scheme is known as "role-based access control."

The proposed HIPAA security regulation, at § 142.308(c)(1)(i)(B), mandates the use of either

1. user-based access control,

2. role-based access control or

3. context-based access control.

For most applications, either user-based or role-based schemes would provide the needed protections, and the choice between the two would be made on the basis of administrative ease. The overwhelming complexity of the HIPAA privacy and security requirements, however, dictates that we consider the third option for health-related applications.

Context-Based Access Control

A context-based access control scheme begins with the protection afforded by either a user-based or role-based access control design and takes it one step further. Access control decisions in a user-based or role-based framework answer questions similar to, "Should this person (or a person who performs this job function) be allowed to access this type of data?" The equivalent context-based question would be, "Should this person (or a person who performs this job function) be allowed to access this type of data *as it applies to this particular patient*?" Context-based access control takes into account the person attempting to access the data, the type of data being accessed and the *context* of the transaction in which the access attempt is made.

Relational database products contain a number of built-in features that make context-based access control applications possible. The simplest of these is the VIEW. A VIEW is a representation of a table (or group of tables) that embodies logical row and column selection criteria. It would be possible, for example, to construct a VIEW based on a health insurance subscriber table that included data limited to those subscribers covered by a particular health plan. By granting a claims-adjuster access to such a VIEW rather than to the table as a whole, access is restricted to that information needed to perform the necessary claims-processing function. This fits nicely with the HIPAA requirement spelled out in § 164.514(d)(2) that employee access privileges should be limited to that needed to accomplish their assigned functions.

As one may well imagine, the use of table views to implement a context-based access control model is necessarily limited. For one thing, to define separate views of a large number of tables for each employee could introduce a significant administrative burden. More importantly the rules used to construct a view are necessarily limited to the logic that can be expressed by a single SQL query (although people who really know SQL can get a lot of functionality into a few lines). To deal with the complexities of HIPAA compliance, more versatile tools are needed. Fortunately, we have one more trick up our sleeve – the *stored procedure*.

Stored procedures are compact bundles of program logic that are stored as part of the database along with table and view definitions. The EXECUTE privilege on stored procedures may be granted to users or roles just as access privileges are granted to tables and views. The original purpose of stored procedures was to boost database performance, which they do well; but stored procedures are also extremely valuable to developers who have to implement complex security requirements. Stored procedures may contain one or more SQL statements that interrogate or manipulate tables. They may also perform logic of any complexity based on the contents of tables and on the values of arguments passed when the procedure is invoked. In essence, a stored procedure allows an otherwise unprivileged user to "borrow" privileges temporarily for a very limited purpose.

The Proposed Model

With the foregoing in mind, here are the core principles on which the proposed context-based access control model is based:

1. All PHI is maintained in relational database tables.

2. Access to PHI tables is allowed only via stored procedures.

3. Each stored procedure contains context-based access control logic.

Tables containing PHI would be constructed as follows:

```
COLUMN 1:  primary-key
COLUMN 2:  subject-individual-key
COLUMN 3:  data (diagnosis codes, etc.)
```

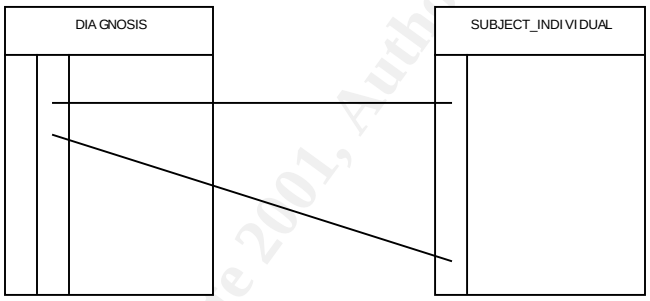
```
...
COLUMN N: (additional data as needed)
```

The first column, the primary key, is a number that uniquely identifies a row. Typically, a database administrator would generate sequence numbers (an ascending sequence of integers) to be used as primary keys. The second column is a *foreign key*, a number that represents a primary key in a different table, in this case a subject individual table.

The SUBJECT_INDIVIDUAL table would be laid out somewhat along the lines of:

```
COLUMN 1:  primary-key
COLUMN 2:  patient-name
COLUMN 3:  ssn (or similar data)
...
COLUMN N: (additional data as needed)
```

The following illustration shows the relationship between the two tables:



The rows of the DIAGNOSIS table are connected to the rows of the SUBJECT_INDIVIDUAL table by a matching key value. The horizontal (and slightly diagonal) lines that connect the two tables are meant to indicate that the value of a key in a particular row is equal to the value of a key in a different row in the other table. A simple query that retrieves information on a particular patient might look like:

```
SELECT  A.patient-name, B.diagnosis-code, B.encounter-date
FROM    SUBJECT_INDIVIDUAL A, DIAGNOSIS B
WHERE   A.patient-name = "JOHN SMITH" AND
        A.primary-key = B.subject-individual-key
```

The "select clause" specifies the particular data items (i.e. columns) of interest; the "from clause" specifies the tables to be interrogated; and the "where clause" specifies two *predicates*. The requirement that data be retrieved for a particular patient is known as the *selection predicate* and

the requirement that the keys match is known as the *join predicate*. Any number of predicates may be listed in the "where clause" according to any combinatorial logic required.

Now we must consider the question of whether or not such a query should be allowed to execute. To begin to answer this question we must posit the existence of two additional tables: a USER table and an AFFILIATION table.

The USER table contains rows for each user who is allowed to log in to the system and execute stored procedures. A USER table might be laid out as follows:

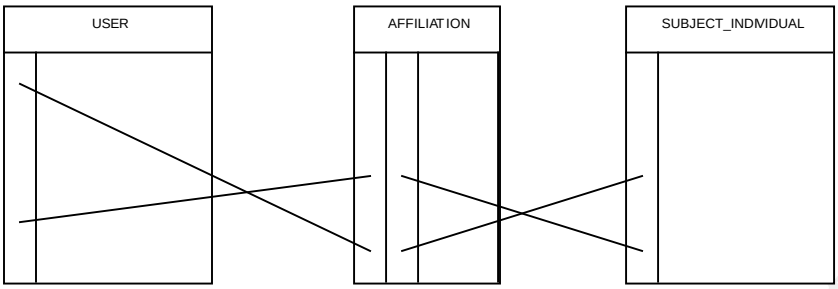
```
COLUMN 1:  primary-key
COLUMN 2:  user-name
COLUMN 3:  contact-info
COLUMN 4:  secondary-authentication-info
...
COLUMN N: (additional data as needed)
```

Secondary authentication information would contain something that a support desk person might use in case the user forgot his or her password (mother's maiden name, etc.). Note that the primary authentication mechanism is the password checking (or biometric checking, token checking, etc.) done by the database engine or the operating system. Similarly, the database engine checks stored procedure execution privileges. No authentication or access control information is stored in application tables and no security functions are performed by application code (this is important).

The AFFILIATION table would be constructed along the lines of:

```
COLUMN 1:  user-key
COLUMN 2:  subject-individual-key
COLUMN 3:  affiliation-type
COLUMN 4:  start-date
COLUMN 5:  end-date
...
COLUMN N: (additional data as needed)
```

Note that the AFFILIATION table does not have a primary-key. The AFFILIATION table is known as an *association* table and is used to establish a *many-to-many* relationship between the USER and SUBJECT_INDIVIDUAL tables. This is shown in the following illustration:



In the model, access control decisions are made primarily on the basis of the existence (or lack thereof) of an affiliation (i.e. a row in the AFFILIATION table) between a user and a subject individual.

An example of such an affiliation would be a treatment relationship between a physician and a patient. After a new patient has reviewed the notice of information practices and has signed the consent for the physician to use and disclose PHI for treatment, payment and healthcare operations, a medical records administrator would add an appropriate row to the AFFILIATION table. This addition would be done via a stored procedure (as is all table manipulation in this design). The administrator would most likely key all of the relevant information into the appropriate medical records application screen, and the underlying application would call the appropriate ADD_PATIENT (or similar) procedure.

The basic structure of a stored procedure declaration (depending somewhat on the particular database vendor) is:

```
CREATE PROCEDURE PROCNAME(arguments) AS

     Variable declarations

     BEGIN

          Procedural logic and SQL statements

     END;
```

A simple context-based access control check that might be done to establish the existence of a physician-patient relationship could be structured as follows:

```
CREATE PROCEDURE PATIENT_CHECK(patient VARCHAR)
RETURN BOOLEAN

rem ----------------------------------------
rem
```

```
rem This procedure checks that a treatment
rem relationship exists between the user and a
rem specific patient.
rem
rem ----------------------------------------


     allow BOOLEAN;
     rowcount INT;
     doctor-key INT;
     patient-key INT;
     relationship-type INT;

     BEGIN

rem Get the key for the current user.

          SELECT primary-key INTO doctor-key
          FROM USER
          WHERE user-name = user;

rem Get the key for the patient in question.

          SELECT primary-key INTO patient-key
          FROM SUBJECT_INDIVIDUAL
          WHERE patient-name = patient;

rem Look up the "treatment" type code.

          SELECT primary-key INTO relationship-type
          FROM RELATIONSHIPS
          WHERE relationship-name = "TREATMENT";

rem Get a count of affiliation rows of this type.

          SELECT COUNT(*) into rowcount
          FROM AFFILIATION
          WHERE user-key = doctor-key                AND
                subject-individual-key = patient-key AND
                affiliation-type = relationship-type;

rem Allow the access if there is at least one such row.

          IF (rowcount > 0) {
                allow = TRUE;
          } ELSE {
                allow = FALSE;
```

```
        }

        RETURN(allow);

END;
```

Note that the precise syntax will vary from one implementation to another. This example is intended to show the logic flow only. (Don't try to execute it.)

The key to this simple context-based access control example is that access will be allowed if the proper affiliation exists between the user and the particular patient in question. A "real-world" access control check would probably not be this simple. For one thing, treatment relationships change. A physician may be allowed to access historical data concerning a past patient but be denied access to current data for the same patient. This is the reason why start and end date columns are included in the AFFILIATION table. To support this type of requirement, we would need to add a DATE_CHECK procedure.

This example illustrates the essential workings of the context-based access control model. Prior to any access to PHI tables, one or more security checks must be done. The logic to perform the security checks may be included in the stored procedure responsible for the accomplishing the data access or it may be a separate stored procedure that is called when needed. The primary check establishes whether or not a relationship exists between the user and the subject individual that would allow the access. Several relationship types are possible, for example:

- **Identity** – The user and the subject individual are the same person.
- **Surrogacy** – The user has the authority to act on the subject individual's behalf.
- **Treatment** – A doctor/patient relationship exists between the user and the subject individual.
- **Employment** – The user sponsors a health plan under which the subject individual is covered.
- **Underwriting** – The user represents an insurer to which the subject individual submits claims.

Secondary access control checks that address access limitations based on exception cases or special circumstances may be performed by additional stored procedures designed for each purpose.

This model is particularly well suited to address a particularly onerous HIPAA requirement: the need to provide an accounting of disclosures of PHI as provided for in § 164.528(a)(1). Wherever an authorization is needed to disclose PHI the subject individual may request an accounting of all such disclosures that have been made in the last six years. This accounting includes certain mandatory components including the identities of the person making the

disclosure and the person to whom the disclosure is made; the date of the disclosure; the type of information disclosed; and the reason for the disclosure. The burden is not alleviated if very few people (or none, for that matter) make such a request because by § 164.528(d)(1) the covered entity is required to keep this disclosure accounting information on file "just in case."

The context-based access control model comes to the rescue by recording disclosure accounting information in a database table designed for that purpose. To accomplish this we need only add logic to the relevant stored procedures to INSERT such information at the completion of the access request. The advantage of capturing this kind of information as part of the transaction is that no manual effort is required. If a disclosure accounting request is received a simple procedure can retrieve the desired information. If no such request is received the only cost is limited to the disk space consumed.

Before leaving the description of the model, I would like to reemphasize the dependency of the model on core operating system and database security controls. For the model to be properly implemented all application tables, and the stored procedures used to access the tables, should be owned by a userid for which the log-in privilege has been disabled. Users should be granted access to the procedures needed to accomplish their assigned job functions but no one should be granted access to the tables themselves. Finally, all invocations of the stored procedures should be done in the context of valid user database sessions.

A popular design scheme for client-server database applications is to have a single database user perform all database accesses on behalf of all clients. Developers give several reasons for employing this technique:

- To improve performance.
- To interface with a particular application package.
- "I don't know how to do it any other way."

While this may be acceptable for certain applications; used in connection with the context-based access control model it is wrong in at least three important respects:

1. **It bypasses tested security mechanisms.** – Anyone who follows any of the popular advisory services such as the Internet Storm Center appreciates how incredibly difficult it is to get security right. Every day we discover new vulnerabilities and new exploits. To discard well-tested security tools in favor of "home-grown" solutions is foolhardy in the extreme.

2. **It spreads the TCB[9] out over a wide area.** – Proper security functionality can only be validated if the Trusted Computing Base (the collection of components responsible for security functionality) is contained in a section of code small enough to be reviewed and understood. Allowing the security mechanisms to be located anywhere in application code makes the review process to difficult to do properly.

3. **It makes critical code vulnerable.** – Whenever security code is touched, the application should be subjected to the most severe testing possible. For this reason, the security component should be kept separate for the rest of the application and changed rarely. Application code is changed all of the time. If the security and application modules are intermixed it is only a matter of time before an application fix introduces a security defect.

Where Do We Go From Here?

To extend the concepts presented here, it would be useful to examine the HIPAA privacy and security regulations in more detail and to identify the core set of procedural algorithms that would be applicable to a wide variety of covered entities. Once this was accomplished a full reference implementation could be built and published. I will be happy to share my ideas and insights with anyone whose chooses to explore this technology further.

Sources

Brittin, Alexander; Pashkoff, Dana B.; and Tedesco, John P. The HIPAA Handbook: What Your Organization Should Know About the Proposed Federal Security Standards. URAC, August 2000.

Health Privacy Project. "Myths and Facts About the Health Privacy Regulation." 5 March 2001. URL: http://www.healthprivacy.org/info-url_nocat2303/info-url_nocat_show.htm?doc_id=53028 (10 July 2001).

The Cato Institute. "The New Medical Privacy Regulations: Will They Protect Our Most Personal Information?" The Cato Institute Policy Forum. 13 March 2001. URL: http://www.cato.org//events/transcripts/010215et.pdf (10 July 2001).

Theriault, Marlene and Heney, William. Oracle Security. O'Reilly, October 1998.

United States Department of Health and Human Services. "Final Privacy Rule." Administrative Simplification. 7 July 2001. URL: http://aspe.os.dhhs.gov/admnsimp/final/PvcTxt01.htm (10 July 2001).

United States Department of Health and Human Services. "HHS Guidance on the Final Privacy Rule." Administrative Simplification. 6 July 2001. URL: http://aspe.os.dhhs.gov/admnsimp/final/pvcguide1.htm(10 July 2001).

United States Department of Health and Human Services. "Proposed Security Rule." Administrative Simplification. 11 April 1999. URL: http://aspe.os.dhhs.gov/admnsimp/nprm/seclist.htm (10 July 2001).

United States Department of Health and Human Services. "Public Law 104-191." Administrative Simplification. 21 August 1996. URL: http://aspe.os.dhhs.gov/admnsimp/nprm/seclist.htm (10 July 2001).

[1] The complete text of all HIPAA Administrative Simplification regulations (i.e. transaction and code set standards, privacy standards and the proposed security rule) can be browsed or downloaded at http://aspe.os.dhhs.gov/admnsimp/Index.htm.

[2] This brief historical overview of the evolution of the HIPAA privacy and security regulations was gleaned from several presentations done at HIPAA Summit West, June 20-22, 2001 in San Francisco. Downloadable presentation materials from these sessions can be found at http://www.hipaasummit.com/past3/agenda/index.html.

[3] The full text of the agreement is published in the Federal Register, Volume 65, beginning on page 45666. The Federal Register may be accessed online at http://www.access.gpo.gov/su_docs/aces/aces140.html.

[4] The complete text of For the Record can be viewed online at http://www.nap.edu/readingroom/books/for/.

[5] Grove, Andrew S. Only the Paranoid Survive. Bantam Doubleday, 1996.

[6] An explanation of this notation is in order: All of the HIPAA regulatory text is part of Title 45 of the Code of Federal Regulations (CFR) so "45 CFR" is omitted in the citations. § 164.506(a)(1) means "45 CFR Part 164, Section 506, paragraph (a), sub-paragraph (1)." The Code of Federal Regulations can be accessed online at http://www4.law.cornell.edu/cfr/.

[7] A comprehensive bibliography of Codd's research can be found online at http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/c/Codd:E%3D_F%3D.html.

[8] To learn more about SQL syntax take the free INT Media Group online course at http://www.sqlcourse.com/.

[9] The "Trusted Computing Base" concept is fully explained the "Orange Book" (i.e. Trusted Computer System Evaluation Criteria). This and most of the other titles in the "Rainbow Series" can be downloaded at http://www.radium.ncsc.mil/tpep/library/rainbow/index.html.

Last Updated: August 16th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced