CELEBRATING 25 YEARS OF EXCELLENCE

HIPAA Compliant **Developer** Guide

HIPAA Audited

GDPR READY

AICPA SOC
aicpa.org/soc4so
SOC for Service Organizations | Service Organizations

HITECH Audited

atlantic.net

Secure Cloud Services
Managed & Compliant Infrastructure

888-618-DATA (3282)
sales@atlantic.net
www.atlantic.net

Table of Contents

The intended audience of this ebook is those concerned with software development across the entire spectrum. That includes readers who are taking on their first project that involves the need for a federally compliant (i.e., legal) healthcare environment, as well as people who have built medical apps in the past and just need a refresher on core concepts and terminology. It includes developers and those who are managing development teams.

The laws: HIPAA & HITECH

Healthcare is a tricky field when it comes to development because there is an additional layer of concern beyond what is needed for the typical website: federal compliance. You need to meet the regulations mandated both by HIPAA and by HITECH.

To understand the very basic function of these two laws, HIPAA was passed in 1996 to allow people to continue coverage when leaving a job or in similar scenarios (portability) and to establish guidelines for healthcare organizations related to safeguard protected health information, or PHI (accountability). HITECH, contained within the American Recovery and Reinvestment Act of 2009 (ARRA), updated some of the HIPAA stipulations and stimulated (through incentives) the adoption of electronic records.

Through the Young Lawyers Division of the American Bar Association[1], Kara J. Johnson explained that the core concern of HITECH was to make it easier for authorized providers and other organizations to access your healthcare records. "[H]owever, because of increased concerns associated with electronic records containing protected health information ('PHI')," she added, "heightened enforcement and sanctions provisions in the... [HIPAA] Privacy and Security Rules were implemented as well."

HITECH is the basis of the HIPAA Final Rule[2], otherwise known as the HIPAA Omnibus Rule[3] or the HIPAA Omnibus Final Rule[4]. The standard, which went into effect in 2013, expanded direct responsibility under the law to third parties that handle PHI on behalf of healthcare organizations.

The parties: covered entities & business associates

The two types of organizations that need to meet HIPAA compliance are called covered entities (CEs) and business associates (BAs). While a

CE must be within one of three categories specified by the HHS – healthcare plans (e.g., insurance carriers), providers (e.g., hospitals), and data clearinghouses – a BA is any company that comes into contact with a healthcare organization's PHI. Examples of common business associates are shredding companies, web hosting firms, and attorneys. Think of any type of service that might come into contact with its clients' records, and you get the idea of a business associate.

The rules: Privacy, Security, and Breach Notification

Three of the core requirements of HIPAA that are often described in the same breath[5] are the Privacy, Security[6], and Breach Notification Rules[7]. All of these standards are within the HIPAA Administrative Simplification Provisions[8].

The HIPAA Privacy Rule is a regulation that must be met by all healthcare providers, plans, and data clearinghouses in the United States – as well as by their business associates – in their treatment of protected health information. It creates national standards that should be used to safeguard electronic health records and other types of confidential medical information. While a huge amount of focus today is put on electronic PHI (ePHI), protected health information must be safeguarded in all its forms and ways it can be communicated, extending to paper, film, and speech.

In order to defend against potential threats to the privacy of these highly sensitive files, the organizations that are regulated by HIPAA have to take action. First, they must actually set up technical protections for the ePHI (which is the core focus of the Security Rule). Second, CEs and BAs must set up controls, as indicated within policy and procedure documents, to prevent any unauthorized use or disclosure (i.e., anything that goes beyond your written agreement with the patient).

The Privacy Rule also established

rights of patients within the United States related to health records. Beyond the broad right to protection of their records, US-based patients have the specific right of access; they can acquire and assess any or all of their records. They also have the right to have any mistakes within the information rectified.

As established above in the need for written agreement, another standard set forth within the Privacy Rule is that patients have to be given a notice of any ways that PHI might be disclosed and used, along with basic information on the responsibilities of the CE and rights of the individual.

To actually enter the vortex and look at this requirement, see 45 CFR Part 160 and Subparts A and E of Part 164 in the HIPAA Administrative Simplification provisions[9]. You can also potentially make use of the tools and guidance provided through the HHS Privacy Rule Page[10].

The HIPAA Security Rule created guidelines with which organizations must safeguard the availability, integrity, and confidentiality of ePHI that is transmitted, maintained, received, or created by a CE or BA. This regulation has been in effect since April 20, 2005, for larger organizations and since April 20, 2006, for smaller organizations.

The Security Rule made it necessary for any organizations handling ePHI to set up defenses in three categories – called administrative, technical, and physical safeguards. The HIPAA regulations established broad needs for healthcare records without usually giving specific directions in terms of technologies, protocols, or methods. When you create launch specifications for a HIPAA-compliant environment, you should include a greater idea of how you intend to meet the demands of HIPAA; to meet compliance, the choices you make should be reasonable and based on industry best practices.

If you really want to dig into the Security Rule, you can find it in 45 CFR Part

160, as well as in Part 164, Subparts A and C, within the Administrative Simplification provisions[11]. The tools and resources organized on the HHS Security Rule Page[12] may also be useful.

The Breach Notification Rule[13] is one of the other key regulatory concerns for covered entities and business associates. One thing should be clear and will help to explain the relationship between HIPAA and other core federal healthcare law. The Breach Notification Rule was introduced within HIPAA and updated within HITECH.

The Breach Notification Rule established that healthcare organizations had to let any patients know right away when their records had been compromised; HITECH expanded this same requirement to business associates. Scope of a breach impacts the compliant reporting process. When the number of records breached is greater than 500, the breached organization should send notifications (beyond those directly to patients) to the Secretary of the HHS and to the media. With any breaches that are considered small (fewer than 500 records) should be sent to the Secretary of the HHS once per year. Plus, there is the issue of breach notification communication occurring properly between business associates and covered entities. When a business associate experiences a compromise to the PHI it handles, it must promptly let the covered entity know, in writing, of the incident.

To look over the regulations within the HIPAA regulations, you can find it in 45 CFR, 400-414 of Part 164 within the Administrative Simplification provisions[14]. The inclusion of business associates as responsible parties related to the need to communicate breaches is described within HITECH[15], section 13407. You can get additional guidance and assistance through the information and resources on the HHS's Breach Notification Rule Page[16].

Note that related to all of these other rules and other important elements of HIPAA and HITECH, it is neces-

sary that you provide training to all your personnel – which is also in the best interests of your organization in terms of avoiding all the negative consequences of a breach.

Checklist: HIPAA mobile app security

Development requirements will be a bit different depending on what type of environment is involved – such as a website, mobile app, or web app. There is not enough space in this ebook for comprehensive coverage of steps for all scenarios; however, it helps to get a bit more specific. To that end, we will drill more deeply with checklists for the development of HIPAA-compliant mobile applications and web applications, upping the ante with the granularity in the second of the two. This information should help with development, and you certainly want to modify these parameters to suit your circumstances.

First, to achieve HIPAA-compliant mobile app security, several steps are key, as indicated by mobile app security software firm NowSecure[17]. The checklist is organized into five sections:

Know what your part is in ensuring HIPAA compliance

- You should know the data protections that are needed within healthcare software. A security professional should look over the design to ensure it suits the scenario. You do not need to become an expert on security or healthcare law to create the app, provided you get advice from competent parties.
- Consider the ways that the application will be used. Think about the types of data the software will be processing and the storage environment. Protecting in-transit and at-rest ePHI is key to maintaining compliance, so consider encryption and other security methods of all system components.
- Be aware if there are other laws that are pertinent to the application. You can use the Mobile Health Apps Interactive Tool[18] from the Federal

Trade Commission (FTC) to automatically determine the regulations that might apply.

Reduce your risk

- Think in terms of minimizing the data that you are presenting, accessing, or storing. There is no reason to gather date-of-birth, for instance, unless it really is needed. You should have defined purposes for all personal data you collect.
- Develop an easily understandable privacy policy, and use it consistently. A privacy policy is important with all applications you develop, but it is particularly key to healthcare.
- One of the best ways to reduce your security vulnerability is not to store data that is highly sensitive. Whenever you do not need to store protected health information, do not do it. It is key to ensure that any data you remove from the system is completely wiped. It isn't impossible to ensure SSD/flash data is completely erased, but it can be difficult if not in control of the systems. By allowing a HIPAA-compliant hosting company to manage the data, the erasure and disposal of sensitive information is taken care of with processes that are already tested and in place.
- It is crucial to consider secure transmission and storage of data when you use cloud technology. For cloud service providers or any other outside contractors, you will need a signed business associate agreement (BAA), as indicated within the HHS's Guidance on HIPAA & Cloud Computing[19].
- Be aware of data related to geolocation. You should not be getting specific about the location of a particular individual if you can help it – which, according to the HHS, means not getting any more specific to exact location than the US state. You might be handling information that does not seem that important or sensitive but that is turned into PHI because of geolocation.

Send and store data using appropriate technical safeguards

- Encryption is a standard data security method. App Transport

Security (ATS) should be included so that the application has to use HTTPS protocol to communicate with servers rather than the standard HTTP – a method that ensures encryption when data is in motion.

- There are many types of security tools available, and they should be used to keep data safe both when it is being transmitted and stored. Encryption methods allow you to verify data as well, which is another key point within the regulations.
- Consider any text messages, and be certain there is no health data within them since SMS and MMS are built without encryption.
- Do not come up with your own encryption algorithm when you are encrypting local storage. Instead, implement protocols that are already widely used.

Set up security protections for the application itself

- You want to have a timeout period established for any local session so that people need to reauthenticate when they are not actively using the app. The determination of a good session timeout should be based on your use case for the app.
- Avoid push notifications. The problem with push notifications is that it is possible they reach the device, and someone other than the patient views it.
- You do not want to allow any leakage of your health information into areas that tend to have poor protections, such as log files. Within an Android device, access is typically problematic and leads to heightened risk because the permissions are often not tightly controlled.
- Use a comprehensive set of security methods that are considered industry standards.

Perform security testing

- Security testing should be conducted to verify that everything is protecting the application properly in both static and dynamic contexts.
- Penetration testing through a third party can be a good idea, particularly if the provider has HIPAA expertise.

Checklist: HIPAA web app security

There is a checklist for building HIPAA-compliant web applications provided by the Open Web Application Security Project (OWASP)[20]. There is a bit of overlap with the above checklist. However, as stated above, this checklist takes a different approach in getting very detailed with the steps that are advised. It is organized into 11 sections:

Gather information

Assess the rendered site

- Look over the site.
- Use a spider to mine your data[21] and check for any hidden or missing elements.
- Check for data leakage via webserver metafiles, such as .DS_Store, sitemap.xml, or robots.txt.
- Verify that you cannot be accessed through search by checking the caches of prominent engines.
- Verify that content does not differ based on the user, such as a search engine spider vs. mobile.
- Confirm that there is no data leakage[22] via webpage metadata and comments.

Assess development

- Verify the framework used in the application's design.
- Fingerprinting of the app.
- Check the implemented technologies.
- Assess user roles.
- Determine points of entry.
- Be aware of client-side scripts.
- Determine all channels or versions of delivery – such as mobile app, mobile web, and web.

Assess the platform and hosting

- Determine any content that is served by independent parties.
- Assess all ports and hostnames used.
- Figure out what applications are co-hosted.
- Verify all web services

Manage configuration

- See what administrative or application URLs might be implemented that are too common to be

secure.

- See what files are unreferenced, backups, or old.
- Confirm all supported HTTP techniques and that Cross Site Tracing (XST)[23] is being prevented
- Verify the processing of file extensions.
- Assess your policy to control rich internet application (RIA) cross domain access.
- Confirm that there are secure HTTP headers in place.
- Check policies for all technologies (such as robots, Silverlight, and Flash).
- Determine if there is any confidential information, such as login credentials or API keys, within client-side script.

Confirm transmission security

Look at encryption and protocols used

- See what the key length is, the algorithms that are used, and the SSL version.
- Verify that you have valid digital certificates.
- Confirm that HTTPS is used any time that usernames or passwords are sent.
- Determine that HTTPS is implemented whenever the login form is sent.
- Be sure that HTTPS is in place for delivery of all session tokens.
- Verify the implementation of HTTP Strict Transport Security (HSTS).
- Confirm that requests cannot be forged.
- Assess HTML5 web messaging.
- Confirm the use of CORS, also applicable to HTML5.

Assess representational state transfer (REST) and web services

- Verify REST implementation.
- Check for any problems with web services.

Verify authentication

Determine functionality of the app password

- Confirm that the password quality rules suffice.
- Verify the proper working of remember me.
- Check that recovery and reset

options function correctly.

- Check that a password can be changed correctly.
- Make sure the CAPTCHA is functioning properly.
- Confirm proper working order of multi-factor authentication (MFA).
- Verify that logout functions correctly.
- Check for any default logins
- Verify proper functionality of notifications related to password changes and account lockouts.
- Be certain that your authentication is consistent throughout applications with alternative channels and shared authentication schema/SSO.
- Determine question/answer issues that represent weak security.

Assess other functionality concerns with authentication

- Check to see if nefarious parties can successfully enumerate users.
- Determine if authentication bypass can occur.
- Verify your defenses against brute force attacks.
- Confirm that encryption on channels through which credentials are travelling is functional.
- Verify HTTP cache management (such as Max-age, Expires, and Pragma).
- Determine proper working order of any authentication history that is user-accessible.

Manage the session

- Determine whether tokens in cookies, token in URL, or another session management method is in place.
- Look for cookie flags with session tokens (both secure and HTTP).
- Confirm the max-age and expiration related to duration of session cookies.
- Following a maximum lifetime, determine that session termination occurs.
- Following a relative timeout, verify that the session terminates.
- Following a logout, verify that the session terminates.
- Check if it is possible to open more than one simultaneous session per user.

- Gauge the randomness of session cookies.
- Confirm that when login, logout, and role changes occur, a new session token is created.
- When there is shared session management for the app, verify that session management is consistently applied.
- Determine if session puzzling is occurring.
- Assess to ensure protection from clickjacking and cross-site request forgery (CSRF)[24].

Verify that authorization is occurring properly

- Check on path traversal.
- Gauge the system for missing authorization.
- See if insecure direct object references[25] are present.
- See if privilege escalation is occurring, which means you need stronger vertical access control.
- See if there are any issues with horizontal access control.

Ensure your cryptography is working correctly

- Gauge for any instance of weak algorithms.
- See if algorithms are being utilized correctly based on the appropriate context.
- Assess the randomness functions within your system.
- See that salting is occurring as intended.
- See that encryption is actually occurring to the data.

Confirm that data is validated correctly

Test for various types of injection

- SQL
- HTML
- LDAP
- ORM
- XML
- XXE
- SSI
- XPath
- XQuery
- IMAP/SMTP
- Code
- Expression language

- Command
- NoSQL

Perform additional validation tests

- See if reflected cross site scripting is occurring.
- Check for the occurrence of stored cross site scripting.
- Verify that DOM based cross site scripting is not taking place.
- Gauge the environment for cross site flashing.
- See if overflow is occurring.
- Check for any format string issues.
- Determine if any incubated weaknesses are present.
- See if smuggling or splitting of HTTP is occurring.
- Check if there is verb tampering with the HTTP.
- See if open redirection is occurring.
- Verify that remote file inclusion is not occurring.
- Check to see that local file inclusion is not taking place.
- See that the validation rules for the server-side and client-side are consistent.
- Check to see if parameter pollution is occurring with HTTP.
- See if auto-binding is taking place.
- Gauge for Mass Assignment
- See that NULL/Invalid Session Cookie is functioning properly.
- Check to see that the data integrity is maintained.
- See that work flows are not being circumvented[26].
- Verify that you are protected against misuse of the application.
- Confirm that you cannot go beyond the limits of a feature or function.
- Check process timing[27] for consistency.
- Related to HTML5, see if web storage SQL injection is taking place.
- Confirm that the application functions properly when it is offline.

Check for denial of service (DoS) concerns

- Gauge for anti-automation.
- See that account lockout[28] is working properly.
- Verify that SQL wildcard DoS is

not occurring.

- Check to ensure that the system is not vulnerable to HTTP protocol DoS.

Work directly with functions that make you vulnerable

Verify that the uploading of files is secure

- Be certain that only the types of files on your whitelist will upload.
- See that the total file count, upload frequency, and size limits are correctly in place.
- Gauge to see that the file type and contents fit.
- See that anti-virus is implemented for all upload types.
- Verify that malicious files cannot be uploaded.
- Make sure sanitizing is taking place for any problematic filenames.
- Be certain that you cannot get to files you upload through the web root.
- Check to see that the same port/hostname is not serving uploads.
- Make sure that your authorization and authentication systems are being applied to all files you upload.

See if there are issues with payment

- Check both the application and server to see if known issues with configuration or weaknesses are present.
- Check to see if passwords are either guessable or default.

- See if buffer overflows are occurring.
- Check to see if there are any weaknesses that might allow injection.
- Determine if insecure cryptographic storage is present.
- Gauge for insufficient protection of the transport layer.
- See that error handling is proper.
- See if there are any weaknesses present that have a score greater than 4.0 according to CVSS v2[29].
- Determine if there are any problems related to authorization or authentication.
- Gauge the system for CSRF vulnerability.

Verify correct handling of errors

- Gauge the stack traces.
- See that the error codes are working properly.

Rooting your development project in a HIPAA-compliant host

That gives you a basic idea of key HIPAA requirements and terminology, as well as specific checklist elements that are important for mobile and web application development. Using the above guidance, you should be well on your way to a HIPAA-compliant development environment.

Developing websites and applications can always be challenging. Simply concerning oneself with usability, it is always possible to make it better. The same is true with privacy and security – and security requires a particularly in-depth exploration when you are handling ePHI.

Get Help with HIPAA Compliance

HIPAA Compliant Hosting by Atlantic.Net is SOC 2 & SOC 3 certified and HIPAA & HITECH audited, designed to secure and protect critical healthcare data and records. Get a free consultation today! Call 888-618-3282 or review our solutions at https://www.atlantic.net/hi-paa-compliant-hosting/.

References

[1] *https://www.americanbar.org/groups/young_lawyers/publications/the_101_201_practice_series/hitech_101.html*

[2] *https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/combined-regulation-text/omnibus-hipaa-rulemaking/index.html*

[3] *https://www.aaos.org/AAOSNow/2013/Jul/managing/managing4/*

[4] *http://www.physicianspractice.com/compliance/two-essentials-hipaa-omnibus-final-rule-compliance*

[5] *https://www.cms.gov/Outreach-and-Education/Medicare-Learning-Network-MLN/MLNProducts/Downloads/HIPAAPrivacyandSecurityTextOnly.pdf*

[6] *http://www.himss.org/library/interoperability-standards/security-standards*

[7] *https://www.hhs.gov/hipaa/for-professionals/breach-notification/index.html*

[8] *https://www.hhs.gov/sites/default/files/hipaa-simplification-201303.pdf*

[9] *https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf*

[10] *https://www.hhs.gov/hipaa/for-professionals/privacy/index.html*

[11] https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf

[12] https://www.hhs.gov/hipaa/for-professionals/security/index.html

[13] https://www.ama-assn.org/practice-management/hipaa-breach-notification-rule

[14] https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf

[15] https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/understanding/coveredentities/hitechact.pdf

[16] https://www.hhs.gov/hipaa/for-professionals/breach-notification/index.html

[17] https://www.nowsecure.com/blog/2017/03/23/5-vital-tips-developing-hipaa-compliant-mobile-apps-checklist/

[18] https://www.ftc.gov/tips-advice/business-center/guidance/mobile-health-apps-interactive-tool

[19] https://www.hhs.gov/hipaa/for-professionals/special-topics/cloud-computing/index.html

[20] https://www.owasp.org/index.php/Web_Application_Security_Testing_Cheat_Sheet

[21] https://www.owasp.org/index.php/Testing:_Spidering_and_googling

[22] https://www.hipaajournal.com/himss-top-healthcare-security-threats/

[23] https://www.owasp.org/index.php/Cross_Site_Tracing

[24] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[25] https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004)

[26] https://www.owasp.org/index.php/Testing_for_the_Circumvention_of_Work_Flows_(OTG-BUSLOGIC-006)

[27] https://www.owasp.org/index.php/Test_for_Process_Timing_(OTG-BUSLOGIC-004)

[28] https://www.owasp.org/index.php/Testing_for_Weak_lock_out_mechanism_(OTG-AUTHN-003)

[29] https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator