

Test Strategies for HIPAA Compliancy

Why we wrote this paper

If you search for information on testing HIPAA compliant applications, you'll find a number of overviews on HIPAA, but very little information on how to approach testing software to ensure that it will be HIPAA compliant. Macadamian's Test Strategy includes HIPAA compliancy verification as an important part of its initial sanity testing and later full feature testing when we are testing and developing healthcare applications. We decided to publish our test strategies to provide developers and software testers with a practical, hands-on guide to testing for HIPAA compliancy. We hope you find it useful, and please contact us with questions and comments.

What you can expect

For simplicity, we describe HIPAA compliancy testing in five main areas (even though there is some overlap between them):

1. User authentication
2. Information disclosure
3. Audit trail
4. Data transfers
5. Information on correct data use

We also ensure that HIPAA compliancy defects are clearly indicated in our bug tracking system so their fixes are given high priority.

Note that this document only covers major software testing areas related to ensuring HIPAA compliancy, and does not cover other areas such as required physical safeguards like not deploying the software on workstations with publicly viewed screens. Also be aware that testing strategies should depend on the software requirements of the application. Not all testing strategies described in this document are applicable to all applications.

Preparation and Sanity Testing

Roles Matrix

Assuming the application makes use of RBA – Role Based Access (or a similar strategy), the first step in preparing for HIPAA compliancy testing is identifying all the roles present in the system and their expected access levels to all components of the application. This identification is done in communication with the customer who then also approves the risk level associated with each role/component/operation relationship. The risk level is based on information disclosure, frequency of use, chance of error, and impact to the customer if an error occurs in a given component.

The table below is an example of what a Roles Matrix might look like. In each cell:

- “x” indicates that the role has the ability to perform the specified operation (View, Add, Modify, Delete) in the application component.
- the color indicates the privacy or security risk level if there were a defect in this role/component/operation relationship, where:
 - Red = high
 - Yellow = medium
 - Green = low

Roles	Application Components																			
	Appointment Management				Patient Charting				Shift Management				Billing				Reports			
	View	Add	Modify	Delete	View	Add	Modify	Delete	View	Add	Modify	Delete	View	Add	Modify	Delete	View	Add	Modify	Delete
Administrator	x	x	x	x									x				x	x	x	
Nurse	x				x	x	x		x								x	x	x	
Supervisor	x	x	x	x	x	x	x		x	x	x	x	x	x	x		x	x	x	
Doctor	x				x	x	x		x				x				x	x	x	
Accountant									x				x	x	x		x	x	x	
IT Supervisor																	x	x	x	

As you proceed to sanity testing, this chart helps you identify the risk level associated with each relationship, and prioritize testing to ensure problems in high risk areas are found and fixed first.

Sanity Testing

Perform HIPAA-related sanity testing as early as possible in the development cycle to quickly discover any major defects in the application’s HIPAA compliancy.

Sanity testing should cover high-level testing in the following areas:

1. For each High risk role/component/operation relationship (as determined in the Roles Matrix described above), verify the following:

- a. A user of the specified role is able to authenticate successfully and is granted view, add, modify, delete, or no access to the specified application component operation.
 - b. Each action (user authentication, viewing, modifying, etc) performed above is recorded in detail in the audit trail.
2. Verify encryption in the following areas:
- a. EPHI (electronic protected health information) in the database.
 - b. Audit trail entries (as described above).

Test Case Details

To aid in HIPAA compliancy and provide an exact traceable record of what is tested, our test cases are written with very explicit details. The individual steps to be executed are broken down to a low level action (with sample input if applicable) and a specific expected result. This way, test cases can be failed at specific steps, making it easier to write clear defect reports..

For example, here is what a test case for a nurse logging in and viewing a patient's care plan might look like:

Test #1234: Nurse Edit Patient Charting Care Plan

Login as a nurse. View and edit a patient's care plan within his charting records.

Step	Description	Expected Result
1	Start the application.	Login screen appears.
2	Login as a nurse.	Nurse view of application opens.
3	Click Patients link.	A list of only the nurse's patients is displayed.
4	Click on a patient in the list.	The selected patient's Patient Charting record is displayed.
5	Click the Care Plan link.	The patient's care plan is displayed in the correct layout according to the wireframe.
6	Press Edit button.	The Edit Patient Care Plan popup opens with proper layout.
7	Clear the information in the mandatory fields. Press Save.	A message is displayed saying mandatory data is missing or invalid. Asterisks and examples of valid data are beside the offending fields.
8	Enter data that is too long for one of the fields.	Only the maximum length of data is displayed in the field.
9	Enter valid different data into the fields. Press Save.	The Edit Patient Care Plan popup closes and the patient's new care plan is displayed.
10	Click Back to Patient List link.	The patient is still displayed correctly in the list.
11	Click Logout link.	Nurse is logged out of the application.

Areas of In-depth Testing

After you have completed sanity testing on all the High risk relationships, proceed to the same sort of tests around the Medium risk relationships, followed by the Low risk relationships.

In addition to this, more in-depth testing is performed, as broken down into the five overlapping areas.

User Authentication

User authentication to the application is typically one of the following:

1. ownership based, for example ID cards
2. knowledge based, for example user id/password
3. biometric based, for example fingerprint

User authentication testing needs to cover more than just the successful login path for each role as described in the sanity testing section. In addition to this, test the negative path and more intricate test cases such as the following (depending on the method of authentication):

- login failure for each of the following reasons, where applicable:
 - empty user id
 - empty password
 - invalid user id (including case-sensitivity if applicable)
 - invalid password (including case-sensitivity if applicable)
 - expired account
 - blocked account
 - locked out account (after repeatedly failing login x number of times)
- login success after password change
- characteristics of password change itself:
 - cannot reuse previous x passwords
 - forced to change after certain time period
- login idle timeout (user session expires after being idle for a period of time) on both workstations and mobile devices
- login credentials not stored in application memory (if required for security)

For applications which also execute on a mobile device with reduced code, conduct separate testing for user authentication on the actual device.

Again, make use of the Roles Matrix to ensure users are granted the correct access level to all application components.

Information Disclosure

Information disclosure is typically limited with two main strategies (but both may not necessarily be tested if they are not used in the application under test):

1. Role Based Access (RBA) – grouping of users into logical classes which have certain levels of access to specific application components (as described in our Roles Matrix).
2. Patient Allocation (PA) – supervisor assigns patients to a health care provider for a specified period of time (for example a shift in a hospital on a specific floor).

As discussed in the Sanity Testing and User Authentication sections, your testing strategy needs to ensure that information disclosure is limited by RBA (or similar strategy used in the application under test).

When applicable, design test cases to ensure that the PA limitations are also respected, such that application users are only able to view/add/modify/delete patient information that they are supposed to currently have access to, and not able to view/add/modify/delete information for patients that have not been currently allocated to them.

Also under the topic of limiting information disclosure, your test strategy needs to ensure that if the application is uninstalled that all EPHI is completely removed/deleted from the mobile device or system. This includes checking all the locations in which the application may have written data such as install directory (default or custom), Windows Registry, user folder.

As will be discussed in the next section on Audit Trails, also verify that all accesses and attempts to access EPHI are identified and reported.

Audit Trail

After sanity testing, conduct a more thorough analysis of the audit trail.

Depending on the format of the audit trail (for example, it could be log files, database entries, etc) you might use comparison tools to verify that the entries produced match previously constructed benchmarks of expected entries.

Perform the following verification:

1. All expected audit trail entries exist, for every operation performed on the EPHI.
 - Make use of the Roles Matrix to ensure no operations (performed by each role) are missed when writing our detailed test cases that specify exactly which entries are expected.

- Also ensure that entries are created for actions performed on all types of devices including mobile.
2. Each audit trail entry contains the following information:
 - the date and timestamp of the action
 - the user performing the action
 - the access level of the user
 - the patient record id (if applicable) on which the action was performed
 - the action performed or attempted
 - the specific application component from which it was performed (for example, billing versus patient charting)
 - the location or system id (for example, the hospital or clinic's NPI (National Provider Identifier), if applicable) from which the action occurred
 3. Entries conform to the software's clarity requirements, so that the audit trail can be easily followed if necessary for a future investigation.
 4. Entries cannot be removed from the audit trail.
 5. Audit trail can only be viewed by certain user accounts.
 6. All attempts to breach security are recorded in the audit trail in such a way that they stand out for easy identification.
 7. Audit trail is encrypted.

Data Transfers

In addition to the encryption verification performed on the database and audit trail during the sanity testing stage, also use a network analyzer tool (such as Wireshark: <http://www.wireshark.org/>) to ensure EPHI is encrypted during:

1. Data access between all workstations and mobile devices on which the application is installed and the database.
2. Data transfers to an external location.
3. The movement of data to an offline storage location.

If the application under test involves health care business practices like the electronic transfer of claims transactions, remittance advices, enrolment and eligibility transactions, then the test strategy should ensure that the proper EDI (Electronic Data Interchange) X12 formats are used for these processes.

As recommended on page 15 of http://www.wedi.org/snip/public/articles/testing_whitepaper082602.pdf, seven types of HIPAA transaction compliance testing should be performed on the EDI files. To aid in this

validation you may use the Framework EDI tool described here:
http://www.edidev.com/edidev_hipaa_support.htm#Validation%204010.

Information on Correct Data Use

Finally, also verify that the application provides an explanation of correct data use prior to its access. Depending on the application, this could be in the form of verifying there is a help page available for each specific operation that involves EPHI, or testing of a training version of the application that allows users to see how the application works before granting access to real EPHI.

Defect Tracking

Report all defects found that relate to HIPAA compliancy with the keyword “HIPAA” appended to their titles in your defect tracking system. Any that correspond to high risk areas (in the Roles Matrix) should be automatically assigned a *Critical* severity. This ensures that these defects are immediately visible to the Project Leader and are fixed with high priority.

Glossary

Acronym	Full term
EDI	Electronic Data Interchange
EPHI	Electronic Protected Health Information
HIPAA	Health Insurance Portability and Accountability Act
NPI	National Provider Identifier
PA	Patient Allocation
RBA	Role Based Access

Contact Us

For questions or comments about this White Paper, or for more information about our healthcare software design and development services, please contact:

Didier Thizy, Director of Project Management, Healthcare
Macadamian
didier@macadamian.com
+1 613 739-5976 x136