atlantic.net
Secure Cloud Services

BEST PRACTICE FOR BUILDING HIPAA-COMPLIANT PYTHON APPLICATIONS

Python is one of the most popular coding languages used by developers for application creation. With an abundance of tutorials and debug code available online to guide developers, it is a relatively easy language to learn.

Some of the world's most popular applications are written in Python, including the video sharing site YouTube, the social media giant Instagram, and the music streaming service Spotify.

Did You Know?

On the Atlantic.Net Cloud Platform (cloud.atlantic.net), you can spin up a Python client server in seconds. Our one-click Ubuntu application servers have Python3 baked into their automated deployment.

Congress passed the Health Insurance Portability and Accountability Act of 1996 (HIPAA), and the subsequent Security Rule and Privacy Rule amendments of 2003, with the sole intention of securing Protected Health Information (PHI).

The physical, technical, and administrative safeguards introduced by HIPAA legislation apply to healthcare applications written in Python.

DESIGNING HIPAA-COMPLIANT APPS

Any bespoke healthcare applications must be designed within a HIPAA compliance framework, regardless of whether the application is solely used in-house or connected to the public Internet, such as a web application, CRM, or mobile app.

HIPAA compliance must be a core goal in development planning, and the application must meet HIPAA's minimum security and privacy standards to ensure the confidentiality, integrity, and availability of ePHI.

Did You Know?

A Python module called philter-ucsf can obfuscate PHI data exports.

There are two main areas of concern when writing HIPAA-compliant software:

- ✓ Security and Privacy of the Application
- ✓ Security and Privacy of the Infrastructure

While application and infrastructure compliance share synergies, we will focus on ensuring the application meets HIPAA standards.

atlantic.net
Secure Cloud Services

CELEBRATING 25 YEARS OF EXCELLENCE

BEST PRACTICE FOR BUILDING HIPAA-COMPLIANT PYTHON APPLICATIONS

FIVE KEY DESIGN REQUIREMENTS

HIPAA compliant Python applications typically feature the same five core functions. These are:

- ✓ User Controls
- ✓ Access and Authorization Controls
- ✓ Backup and Restore Requirements
- ✓ Software-Defined Disaster Recovery
- ✓ Automatic Logout

USER CONTROLS

Means of control must be built in to protect unauthorized access to the application. Data should only be accessible by authorized personnel who are pre-approved to access ePHI. This includes any in-scope member of the covered entity (typically doctors, nurses, physicians, etc.), thus necessitating numerous administrative controls to document who these authorized users are and what PHI they can access.

Did You Know?

Python has built-in Access Control List capabilities. Extensive documentation is available at https://wiki.python.org/moin/HelpOnAccessControlLists. Some of the key variables are:

acl_hierarchic - True to use hierarchical ACLs
acl_rights_after - ACL that is processed after the on-page/default ACL
acl_rights_before - ACL that is processed before the on-page/default ACL
acl_rights_default - ACL used if no ACL is specified on the page
acl_rights_valid - Valid tokens for right sides of ACL entries

A great example of how this could be achieved might be in an integration with Active Directory or any other LDAP service. However, controls must also prevent superusers, such as application and system administrators, from accessing ePHI. Disabling the root user account is a good start. All user access must be logged via the application, and enabling verbose log shipping enabled is recommended.

Did You Know?

The Active Directory Python tool pyad is an Object-Oriented Active Directory management framework built on ADSI that can integrate with any locally connected domain

atlantic.net
Secure Cloud Services

CELEBRATING 25 YEARS OF EXCELLENCE

BEST PRACTICE FOR BUILDING HIPAA-COMPLIANT PYTHON APPLICATIONS

ACCESS & AUTHORIZATION CONTROLS

Access and authorization controls work together with user authorization with the key difference that application controls are written to restrict access to flagged confidential information. Code must be implemented that prevents the export of data, such as export to an Excel file, PowerShell, or a simple copy & paste. The os.chmod and tempfile.mkstemp() classes are a good starting point.

The available rights that can be used within Python are:

- *read* - users will be able to read a page and download its attachments.
- *write* - users will be able to edit (write) a page and upload attachments.
- *delete* - users will be able to delete a page and its attachments.
- *revert* - users will be able to “revert” a page back to an older revision.
- *admin* - users will be able to change the “#acl” line on a page as well as grant and/or revoke “admin” privileges from others.

A permission-based authority must be present to ensure view and edit controls are granted to the correct teams. If an unauthorized user accesses or changes ePHI, a HIPAA breach has occurred; therefore, strict HTTP request method controls must be enabled to prevent an API requesting to DELETE data. Controls can also be placed directly on patient data which only allow certain doctors to view their specific patient data. Access to the application can be restricted by IP range, and the application can limit access based on the geolocation or privileges of an IP address; for example, users in a hospital subnet of 10.1.2.0/24 might be the only user devices granted access to the Python app, and all other IP addresses are blocked by default.

Did You Know?

While a vast number of geolocation Python modules are available, one of the best is ip2geotools, which can pinpoint the location of an access request.

In any application, database handling plays a major role, as data must be stored for future use. Ensure that the application database is encrypted if processing PHI, either in cache or a database. Encrypt sensitive data in the database using a tool such as Tink, a multi-language, cross-platform, open-source library providing cryptographic APIs that are secure, easy to use correctly, and hard to misuse. Rather than linking data with a username, link sensitive data with a UID, or at least architect your system so data is not traceable back to a single user by first name and/or last name. In the event the database is exploited, this will prevent hackers from being able to see the data listed in the text; the data should be encrypted and need a key to decrypt.

Did You Know?

The Virgil SDK provides end-to-end encryption (E2EE) security APIs for any Python application. You can encrypt communication, securely store data, provide passwordless login, and ensure data integrity.

atlantic.net
Secure Cloud Services

CELEBRATING 25 YEARS OF EXCELLENCE

BEST PRACTICE FOR BUILDING HIPAA-COMPLIANT PYTHON APPLICATIONS

BACKING UP AND RESTORING DATA

Data backup is a key HIPAA technical infrastructure requirement and therefore applies to HIPAA-compliant Python application development. The application should feature a backup and restore function, and it should be able to create offline backups of application data. Use application data consistency checks to ensure PHI data has not been tampered with during the backup/restore process.

If changes are made to application data, automated alerting should trigger a data integrity security alarm, and measures should be put in place to automatically notify the correct support personnel to investigate the alert and, if applicable, resolve the issue.

Restrict export of data from databases at the application layer. Instead, use HIPAA-compliant backup software that has agents capable of backing up in-use SQL or MySQL databases. You must replicate backup data to a secondary US-based data center.

SOFTWARE-DEFINED DISASTER RECOVERY

Disaster Recovery is another key technical safeguard of HIPAA compliance and is referenced directly in HIPAA legislation: "Assess the relative criticality of specific applications and data in support of other contingency plan components."

Many Python functions provide an easy way to create highly-available applications, such as the AMPS module HAClient class. This class offers methods of protection against network, server, and client outages, including:

- ✓ Recovery from temporary disconnects between client and server
- ✓ Failover from one server to another when a server becomes unavailable

Python modules automatically manage the failover and reconnection process and can work seamlessly in an Active-Active or Active-Passive disaster recovery solution. These modules are great for mission-critical workloads and ensure that no messages are lost or duplicated during the failover process.

Did You Know?

The Pypi project Timeout allows developers to introduce timeouts on any Python module or function.