Application Security Assessment Report

Humana Dispensing (HPD) - APP0000729

Version: 1.0

Creation Date: 2025-04-30

Created by: Sidhant Maithani **Date:** 2025-04-30

INDEX

1. Executive Summary

This document provides the results of the Application Security Assessment (ASA) performed for Humana Dispensing (HPD). The assessment focused on identifying design flaws in the application across several security control families. Design flaws have been paired with security requirements to provide a clear path to remediation (i.e., actionable next steps towards resolution). This is the interface between the Humana Pharmacy Dispensing system and any consuming clients for dispensing Orders submitted by the end clients. HPDispensing_InventoryBroker_SVC and HPDispensing_OnPrem_SVC are the only two On-Prem Components, the other are being developed in Azure Cloud.

Executive Summary Section: This document provides the results of the Application Security Assessment (ASA) performed for Humana Dispensing (HPD). The assessment focused on identifying design flaws in the application across several security control families. Design flaws have been paired with security requirements to provide a clear path to remediation (i.e., actionable next steps towards resolution). This is the interface between the Humana Pharmacy Dispensing system and any consuming clients for dispensing Orders submitted by the end clients. HPDispensing_InventoryBroker_SVC and HPDispensing_OnPrem_SVC are the only two On-Prem Components, the other are being developed in Azure Cloud.

2. Overview of Application - Humana Dispensing (HPD)

APP SNOW ID	APP0000729
APP Name	Humana Dispensing (HPD)
APP Description	This is the interface between the Humana Pharmacy Dispensing system and any consuming clients for dispensing Orders submitted by the end clients. HPDispensing_InventoryBroker_SVC and HPDispensing_OnPrem_SVC are the only two On-Prem Components, the other are being developed in Azure Cloud.
Business Criticality	Platinum
Exposure - Internal Vs External	Internal
Sensitive Data	N/A
Application Portfolio/Business Owner	Ashley Day
IT Owner	Falguni Desai
Application Owner	N/A
App Architect	Narasimha Rao Thota
Security Advocate	Tim Shivery
Support Contact	N/A
Other SMEs (Data Protection, Authenticataion SME, etc.,)	N/A

2.1. Application Process Flow

Process flow diagram can be found on this link. TASK3230266

3. Scope and Methodology

3.1. In Scope

The following components of the application are in scope for the assessment:

- HPDispensing_Cloud_OrderTracking_Svc
- HPDispensing_Cloud_OrderResubmission _Svc
- HPDispensing_Cloud_Validator_Svc
- HPDispensing_Cloud_OrderCreate_Svc
- HPDispensing_Cloud_OrderComplete_Svc
- HPDispensing_Cloud_Shipment_Svc

3.2. Out of Scope

3.3. Methodology

The table below describes the methodology used by the assessment team. It has four phases:

- Design Discovery
- Design Assessment
- Flaw Identification
- Document and Recommend

Design Discovery	Design Assessment	Flaw Identification	Document and recommend
1. Review application design and security documents 2. Conduct interviews and workshops with application stakeholders	1. Analyze current design plans and coding practices 2. Verify presence or absence of security requirement implementations.	1. Identify design flaws.	1. Document impact and risk associated with the design flaws. 2. Provide security requirements for remediation.

3.4. Security Requirements

Security requirements ensure application designs follow industry best practices and are aligned with Humana standards. Identifying security flaws during the design phase of an application saves significant resources compared to remediation after development and implementation. The goal of security requirements is to reduce the overall risk of an attack to an acceptable level. These controls may fail, so it is necessary that security design of the application and is assets makes them resilient to an attack through other properly implemented security requirements i.e., a layered security approach.

For this assessment, we looked at 3 types of controls, across 8 application security domains.

1. Type of controls
    a. Preventative
    b. Mitigative
    c. Compensative

4. Design Flaws and Security Requirements

4.1. Design Flaws

The design flaws and risks as well as the implementation security requirements corresponding to each of the domains are reported in the table below

4.1.1.Low: SSL Certificates

Number	APPDFW0002347
Design Flaw	An inventory list of SSL certificates is not created or maintained.
Risk	Failing to maintain an inventory list of digital certificates adds unnecessary complexity to their maintenance and upkeep. In the event an undocumented certificate expires, the maintenance team may not be made aware of it
Assessment Response	Not Implemented
Status	False Positive
Requirement	Create an inventory list of all digital certificates that the application will use. Creating an inventory ensures all digital certificates are accounted for and can be rotated properly prior to expiration.
Mitigation	Create an inventory list of all digital certificates that the application will use. Creating an inventory ensures all digital certificates are accounted for and can be rotated properly prior to expiration. Define a list of all certificates used by the application.

4.2. Security Requirements

Control Family	Control Name	Control Requirement
Access Control	RBAC/ABAC	Access to privileged functions should be restricted to as few users as possible through RBAC or ABAC assignment. Additionally, the number of privileged functions should be the least amount required for the user's assigned role. Example: If there are 5 privileged security groups in an organization and a user's role requires membership to one group for performing their duties then the user must be a member of the one security group.
Access Control	RBAC/ABAC	Access controls must be designed to provide users with security-related responsibilities, two separate accounts: one for non-privileged functions and a separate specifically for privileged functions.
Access Control	RBAC/ABAC	The access control policy must be consistently applied to the

Control Family	Control Name	Control Requirement
		entire application for all user accounts accessing it. Inconsistently enforcing the access control policy may lead to bypassing permissions.
Access Control	RBAC/ABAC	Access to repositories of data (source code, databases, storage accounts) should be limited to the least number of users required.
Access Control	RBAC/ABAC	The access control system must allow user's permissions to be revoked or disabled. Examples include a user transferring from one department to another or a departed employee.
Access Control	Privilege Access Management	Use the Humana standard solution for identity and access management. Using this solution will prevent stale accounts, enforced password rotation, and proper role/permission assignment for accounts.
Access Control	Privilege Access Management	Remove or disable temporary and emergency system/service/privileged accounts after a defined period of time that meets Humana standards.
Access Control	Privilege Access Management	Automatically disable system/service/privileged accounts that are expired, no longer in use, inactive after a defined period of time that meets Humana Standards.
Access Control	Privilege Access Management	An automated method is not used to audit account events: creation, modification, enabling, disabling, and removal of permissions.
Access Control	Privilege Access Management	User/system/service/privileged accounts are reviewed on a regular basis for changes in permissions. Example: A user transfer's from the server administration department to the accounting department. The user's account permissions must be updated to remove access to server administration responsibilities.
Access Control	Privilege Access Management	Use of non-interactive accounts must align with Humana standards: unique per use case for use with application code or a script , not shared across multiple systems, and password vaulted in CyberArk.
Access Control	Directory Browsing	Disable directory browsing by default to prevent unauthorized access to directories containing sensitive information.
Access Control	Directory Browsing	A list of allowed follow extensions must be defined for each directory where browsing is enabled. Defining an allow list prevents disclosure of hidden or sensitive files that should not be available for viewing or downloading.
Access Control	Least Privilege	The application must not allow access to authenticated pages until after a user successfully authenticates.
Access Control	Least Privilege	User accounts should be granted the minimum permissions required, after specifically requested, and once reviewed by the identity management team.

Control Family	Control Name	Control Requirement
Access Control	Insecure Direct Object Reference	The application must validate access controls for every request to prevent Insecure Direct Object Reference attacks that allow access to unauthorized resources: Example attacks include creating or updating someone else's record, viewing all users' records, or deleting all records. Additionally, using a salted hash value to replace identifiers can make it more difficult for attackers to guess identifiers.
Access Control	Re-authenticate for high-risk transactions	Require a user to re-authenticate using a password or multi-factor authenticator before completing high-risk transactions. Requiring re-authentication ensures high-risk transactions are protected if a privileged account's session is stolen.
Audit and Accountability	Event Logging	The application must implement event logs with information that contains relevant information to assist with troubleshooting/incident response.
Audit and Accountability	Event Logging	The logging mechanism must manage the size of data that is logged and prevent using all of the system drive's capacity.
Audit and Accountability	Event Logging	The application's logging mechanism must filter/sanitize/remove any sensitive data before writing to a log: PII, PHI, PCI, secrets/passwords.
Audit and Accountability	Event Logging	Strict access controls for application logs must be enforced to prevent tampering, destruction, and non-repudiation.
Audit and Accountability	Event Logging	Archived logs must have the same protections and role-based access controls as active logs. The integrity of archived logs must be protected to prevent destruction and tampering of logs that may need referenced at a later date for forensic investigation or other compliance laws.
Audit and Accountability	Event Logging	The application must transmit logs to the Humana Standard SIEM solution for backup and protection.
Audit and Accountability	Event Logging	Application logs must be encrypted in transit to prevent tampering to log information or interception through a man-in-the-middle attack.
Audit and Accountability	Secure Code Exceptions	The application must use a "last resort" to handle all unplanned exceptions. An attack will always begin with a reconnaissance phase in which the attacker will try to gather as much technical information as possible about the target. Unhandled errors can assist an attacker in this initial phase by inadvertently leaking verbose information about the application and underlying host.
Audit and Accountability	Application Logging	Application logging controls should be managed by the application host. Storing the logging controls on the host prevent an attacker from disabling or destroying logs.
Audit and Accountability	Application Logging	Logging controls must not be available for modification in client-side code. If logging controls are present in client-side code then an attacker may be able to reduce or disable application logs.

Control Family	Control Name	Control Requirement
Configuration Management	Security Headers	The application must send responses with the Content-Type and charset headers (using an acceptable character set such as UTF-8) to prevent cross-site scripting and other attacks related to encoded characters. CWE-173: Improper Handling of Alternate Encoding - https://cwe.mitre.org/data/definitions/173.html
Configuration Management	Security Headers	Use the Strict-Transport-Security request header to prevent data from being sent unencrypted via HTTP. Example: Strict-Transport-Security: max-age=31536000; includeSubDomains
Configuration Management	Debug Mode Disabled	Disable all debug or verbose modes for the application server and framework to eliminate debug features, developer consoles, and unintended security disclosures/
Configuration Management	Third Party Content Management	Use a verified sandboxing utility to protect the application from compromised third-party content. Compromised third-party content can lead to unauthorized changes to the application, execution of arbitrary code on client systems, leakage of sensitive information. Request third-party libraries from Humana's Artifactory instance where libraries are scanned using jFrog X-ray.
Configuration Management	Third Party Content Management	Creating an inventory of third-party libraries or content used by the application assists with triaging when critical vulnerabilities are released, expediting the mitigation or update process.
Configuration Management	Custom Error Message	The application must use a verified error handling framework that provide helpful messages to users, but not overly verbose to the point of helping attackers trying to gain information about the application.
Configuration Management	Purged Headers/Responses	The application must only use HTTP headers and response data that is explicitly required for the application to function properly.
Identification and Authentication	User Authentication	Use only approved Humana standards for Identity and Access Management.
Identification and Authentication	User Authentication	Shared accounts must be stored in a Human approved Vault solution, requiring authentication of a user's individual account prior to gaining access to the shared account. Additionally, passwords for shared accounts must not be viewable to the user.
Identification and Authentication	User Authentication	The application must enforce multi-factor authentication for privileged and non-privileged interactive accounts. Multi-factor authentication reduce risk from stolen credentials or brute forced accounts. go/policysource-eip: ➜Standard - EIP - Workforce Identity and Access Management
Identification and Authentication	User Authentication	Use only Humana approved multi-factor authentication solutions.
Identification and Authentication	Deny Access By Default	The application must default to denied access when unplanned authentication errors occur. If authentication errors are

Control Family	Control Name	Control Requirement
		unhandled properly, then the application may be left in an unpredictable state (CWE-280: Improper Handling of Insufficient Permissions or Privileges). Recommendations include the following: -Ensure all exceptions and failed access control checks are handled no matter how unlikely they seem (OWASP Top Ten Proactive Controls C10: Handle all errors and exceptions). Additionally, the application should not try to "correct" a failed check; instead a simple message or HTTP status code will suffice. -Centralize the logic for handling failed access control checks. -Verify the handling of exception and authorization failures. Ensure that such failures, no matter how unusually or unlikely, do not put the software into an unstable state that could lead to authorization bypass.
Identification and Authentication	Password Complexity & MFA	Enforce complexity requirements that meet Humana standards to prevent use of weak or previously leaked passwords for user accounts.
Identification and Authentication	Password Complexity & MFA	Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is breached, the application must require the user to set a new non-breached password.
Identification and Authentication	Password Complexity & MFA	The application must force password expiration according to Humana standards ensuring the same password is not reused. -Password rotation every 90 days (Humana Government Business: 60 days) -Requires password not reused for 850 days
Identification and Authentication	Password Complexity & MFA	The application must use multi-factor authentication to enhance the security of user accounts. Multi-factor authentication (MFA) is by far the best defense against the majority of password-related attacks, including brute-force, credential stuffing and password spraying, with analysis by Microsoft suggesting that it would have stopped 99.9% of account compromises. If a password breach occurs, but MFA is enabled then the username/password combination is not enough to grant access to an application.
Identification and Authentication	Password Complexity & MFA	Secrets for non-user accounts must be rotated on a regular basis that meet or exceed Humana standards. Regular secret rotation reduces the lifetime for a given secret reducing risk if it is exposed to an attacker.
Identification and Authentication	Password Complexity & MFA	The application must use a method protect passwords from truncation. Password truncation protections prevent users from registering with a password that is silently truncated on the

Control Family	Control Name	Control Requirement
		registration form, but the full password is accepted at the login form. Resulting in the user being unable to login to their account. An example of protecting the user from silent password truncation can be as simple as a warning during the registration or password change form notifying the user their password is too long.
Identification and Authentication	Authentication for non-public resources	The application must require authentication for all non-public requests, verifying the requesting identity is authorized for the resource that is being requested and the authentication identifier (e.g., session token) has not expired, been disabled, or revoked.
Identification and Authentication	Authentication for non-public resources	The application must require authentication for all systems accessing sensitive information, verifying the requesting system is authorized for the resource that is being requested and the authentication identifier (e.g., session token) has not expired, been disabled, or revoked.
Identification and Authentication	Account Recovery	Implement a recovery process to prevent account lockout from the application in the event of almost multi-factor authenticator.
Identification and Authentication	Account Recovery	Implement an account recovery mechanism that also establishes a new multi-factor method for the user. Some example solutions are provided below: -Providing the user with a number of single-use recovery codes when they first setup MFA. -Requiring the user to setup multiple types of MFA (such as a digital certificate, OTP core and phone number for SMS), so that they are unlikely to lose access to all of them at once. -Posting a one-use recovery code (or new hardware token) to the user. -Requiring the user contact the support team and having a rigorous process in place to verify their identity. -Requiring another trusted user to vouch for them.
Identification and Authentication	Brute Force Attack Protection	Multi-factor authentication (MFA) is by far the best defense against the majority of password-related attacks, including brute-force, credential stuffing and password spraying, with analysis by Microsoft suggesting that it would have stopped 99.9% of account compromises. go/policysource-eip ➜Policy - EIP - Authentication Codes and Passwords ➜Standard - Multi-factor Authentication Solutions
Identification and Authentication	Brute Force Attack Protection	User accounts must enter a lockout state after a defined number of unsuccessful login attempts. The threshold for unsuccessful attempts must meet Humana Standards:
Identification and Authentication	Brute Force Attack Protection	The application must evaluate login information for user accounts and protect if a suspicious login event is detected.
Identification and Authentication	User Change Password	The application must provide a self-service method to allow users to change their current password. Providing a self-service option will allow users to quickly change credentials if a compromise has occurred and reducing the time it is valid.

Control Family	Control Name	Control Requirement
Identification and Authentication	Limited response for password reset or login error	The application must not provide detailed responses for password reset or login errors i.e., specifically what was erroneous about the attempt: bad password, account does not exist, etc.
Identification and Authentication	Limited response for password reset or login error	Authentication responses, regardless of outcome, must be returned in the same average time to prevent an attacker from enumerating information about an account based on the response time. Example: An attacker could learn if an account exists if the application's login page takes longer to respond, indicating the account does not exist because the query took longer to evaluate all users in the database.
Identification and Authentication	Session Management Controls	Users must have the ability to self-terminate their session on all authenticated pages of the application.
Identification and Authentication	Secure "Forgot Password"	The users current password must not be sent during a "forgot password" event. Instead, a backup code, TOTP, soft token, mobile push mechanism must be used to prevent providing an avenue for an attacker to gain a user's password. If these methods are not available the application may send an email with a time-limited, random, one-time use token (generated using a Cryptographically Secure Pseudo Random Number Generator) to the email previously registered by the user.
Identification and Authentication	Secure "Forgot Password"	Use a recommended side channel to communicate the method of password reset to the user. Using a side-channel prevents an attacker from resetting the victim's password by interacting directly with the application. Examples of side-channels include: email account entered during account registration, backup code generated during account registration, soft token, hardware token.
Identification and Authentication	Secure "Forgot Password"	URL tokens must be configured with the following: -Randomly generated using a Cryptographically Secure Pseudo Random Number Generator -At least 64-bits in length. -Stored securely until expiration or use -Single use and expire after an appropriate period. Applying these configurations will protect the forgot password process from an attacker guessing the correct URL token for performing a password reset.
Identification and Authentication	Secure "Forgot Password"	The application must not lockout an account during a pending password reset event. This can be used to deny access to users with known usernames.
Identification and Authentication	Native Session Management	It is recommended to use the application framework's native session management functionality rather than custom code. Native session management functionality has been robustly tested and used across multiple web environments
Identification and Authentication	Two-factor Authentication	The application should use secure methods for authentication: soft tokens (mobile apps), hard tokens, hardware U2F tokens,

Control Family	Control Name	Control Requirement
		smartcards. If insecure authentication methods, SMS and email, are allowed then limit use to secondary verification (multi-factor) or low-risk transactions. The risk associated with use of SMS or email must be clearly communicated to users.
Identification and Authentication	Secure Password Storage	Hashing is a one-way function that creates a work order of magnitude for recovering the plain text of the password. Avoid using encryption for password storage as the password may be easily recovered if the decryption key is also obtained. Note: This may not be applicable for applications that integrate with AD
Identification and Authentication	Secure Password Storage	A salt is a randomly generated string that is added to each password before hashing. Combining a salt will create a harder to crack hash that must be uniquely recovered per password, thus making it harder to crack the password. Note: This may not be applicable for applications that integrate with AD
Identification and Authentication	Alternative Attack Protection	The application must use the Humana approved access control, authentication, and TLS standards to ensure user accounts are not subject to alternative attacks that allow an attacker to easily change a user's password to gain unauthorized access.
Identification and Authentication	Hardcoded Secrets	The application's source code and source code repositories must not contain passwords, secrets, API keys, seeds, etc. Instead, use a Humana approved vault.
Identification and Authentication	Password Checks	The application must verify user submitted passwords (registration and login process) have not been included in a breached list. This may performed using a local list of top 1,000 or 10,000 common passwords or using an external service. Performing this verification ensures breached passwords are not being actively used by the application. A list of popular breached/common passwords may be found at: https://github.com/danielmiessler/SecLists/ A reputable service for API calls is: https://haveibeenpwned.com/Passwords
Identification and Authentication	Password Strength Meter	The application should provide a password strength meter that can assist user's with determining if they are using a weak password.
Security Assessment and Authorization	Security Review	Threat modeling is a structured approach of identifying and prioritizing potential threats to a system, and determining the value that potential mitigations would have in reducing or neutralizing those threats. https://owasp.org/www-community/Threat_Modeling_Process#stride https://owasp.org/www-community/Threat_Modeling_Process#subjective-model-dread
Security Assessment and Authorization	Security Review	The application must support logic to verify business logic parameters and prevent abuse of the system through relaxed validation. An example scenario is an ecommerce site that does not set limits for quantities of items that may be purchased. An

Control Family	Control Name	Control Requirement
		attacker could complete fraudulent purchases with large quantities of items. This would deny honest customers access to goods and services.
Security Assessment and Authorization	Alerting	The application must be able to generate alerts if automated attacks or unusual activity is detected. Alerting for these types of behavior are critical for timely security response. Scenarios of unusual activity should be identified during threat modeling and alerts configured based on those results. An example scenario to configure alert: receiving 100,000 account registration requests generated from a small list of IP addresses in a five minute period.
Security Assessment and Authorization	Time of Check to Time of Use (TOCTOU)	Review the Common Weakness Enumeration article on TOCTOU race conditions: https://cwe.mitre.org/data/definitions/367.html TOCTOU Definition: The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state.
System and Communications Protection	Data Encryption	Use TLS 1.2+ and Humana approved encryption algorithms to secure data in transit Examples of sensitive data includes PII, PHI, PCI, secrets, API keys, etc.
System and Communications Protection	Data Encryption	Utilize the approved cryptographic implementations to protect data depending on its classification
System and Communications Protection	Secrets Inventory	Create an inventory list of secrets (passwords/cryptographic keys/digital certificates) to ensure all secrets are rotated according to Humana standards. Identifying use of secrets during the design phase also ensures secrets are stored in a Humana approved Vault solution and not stored in plaintext of the host or source code repository.
System and Communications Protection	Cryptographic Algorithm	Use only cryptographic algorithms and modules certified by the National Institute of Standards (NIST) for performing cryptographic operations https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program
System and Communications Protection	Cryptographic Algorithm	Encryption components (initialization vectors/nonces, cipher configuration, and block modes) should be configured properly using Humana approved algorithms. Improper configuration of these encryption elements may lead to a weakness in the encryption scheme and leak sensitive data. https://csrc.nist.gov/Projects/cryptographic-standards-and-guidelines
System and	Cryptographic	Encryption components (initialization vectors/nonces, cipher

Control Family	Control Name	Control Requirement
Communications Protection	Algorithm	configuration, and block modes) should be configured properly using Humana approved algorithms. Improper configuration of these encryption elements may lead to a weakness in the encryption scheme and leak sensitive data. For more information please see: https://cryptobook.nakov.com/symmetric-key-ciphers
System and Communications Protection	Cryptographic Algorithm	Cryptographically Secure Pseudo-Random Number Generators (CSPRNG) are designed to produce a much higher quality of randomness (more strictly, a greater amount of entropy), making them safe to use for security-sensitive functionality.
System and Communications Protection	Sensitive Information Storage	Prevent sensitive data from being stored in cache by enforcing the proper HTTP request headers from the application will prevent caching.
System and Communications Protection	Sensitive Information Storage	The application must use a mechanism to immediately purge sensitive information from memory once the use of the data is no longer necessary. Purging sensitive information from memory when no longer in use prevents memory dumping attacks from a compromised host.
System and Communications Protection	Sensitive Information Storage	Session authenticators must be immediately removed from client storage and invalidated on the server when a user willingly terminates a session.
System and Communications Protection	Resource Utilization Protections	The application must use defensive technologies that protect it from service interruption as a result of a denial of service attack.
System and Communications Protection	Resource Utilization Protections	The application must use a defined quota for resource allocation to prevent unnecessary or excessive provisioning of resources due to abnormal increase in traffic.
System and Communications Protection	Code Signing	Code signing is used to verify the integrity and authenticity of binaries or code components including 3rd party libraries prior to use. Code signing validation for executables, binaries, etc. is not enforced by the application.
System and Communications Protection	Session Identifier Protection	Session identifiers must be generated using a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) with a minimum length of 128-bits (16 bytes). Using a CSPRNG to for generating 128-bit session identifiers ensures a user session cannot be brute forced or predicted by an attacker.
System and Communications Protection	Session Identifier Protection	Implement session ID anomaly detection mechanisms to monitor, detect, and block malicious activity focused on gaining unauthorized access to user accounts. To protect against anomalous session ID behavior, the user's session ID can be bound to client properties such as IP address, User-Agent, or client-based digital certificate. If changes in these values are detected then the application can terminate the session. Additionally, the OWASP AppSensor Project can assist with implementing application layer intrusion: https://owasp.org/www-project-appsensor/

Control Family	Control Name	Control Requirement
System and Communications Protection	Session Identifier Protection	In addition to other relevant information (timestamp, source IP address, HTTP resource requested) the application must log all events for user's session ID. Logging session ID lifecycle events can provide critical information for incident response and forensics.
System and Communications Protection	Session Identifier Protection	Humana architects should determine if multiple simultaneous logons from the same user are allowed from the same or from different client IP addresses.
System and Communications Protection	Session Management Controls	The application must invalidate user sessions immediately upon logout. A long lived session that is not invalidated risks an attacker taking over a user's session.
System and Communications Protection	Session Management Controls	New session identifiers must be generated for new login sessions. Using new session identifiers prevent session takeover from an attacker due to session identifier re-use.
System and Communications Protection	Session Management Controls	The application must validate authentication sessions for every request or use of a secondary authentication factor before executing high risk transactions.
System and Communications Protection	SSL Certificates	Create an inventory list of all digital certificates that the application will use. Creating an inventory ensures all digital certificates are accounted for and can be rotated properly prior to expiration.
System and Communications Protection	SSL Certificates	Digital certificate's must match the assigned domain name of the application to avoid certificate validation errors as a result of a failed TLS handshake.
System and Communications Protection	SSL Certificates	Unexpected certificate expiration can interrupt services. Automate certificate renewals to avoid a disruption of service and ensure cryptographic keys (in this case the digital certificates) are rotated according to Humana policy.
System and Communications Protection	SSL Certificates	Intermediate certificates must be installed correctly for all digital certificates in use by the application. Missing intermediate certificates can cause certificate validation errors and disrupt services.
System and Communications Protection	Client-Side Caching Sensitive Data	Disable client-side caching of sensitive data for autocomplete forms. By disabling the caching of sensitive data (PHI/PII/PCI) for autocomplete forms the user supplied information will not be retrievable by any other users of the client system.
System and Communications Protection	Random Value Generation	The application must use an approved and verified Cryptographically Secure Pseudo Random Number Generator for all random values. This is to prevent patterns or re-use of randomly generated numbers that may allow an attacker to guess or forecast values. List of language-specific CSPRNG: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html#secure-random-number-generation
System and	Secure HTTP	The application must send sensitive data in the HTTP body or

Control Family	Control Name	Control Requirement
Communications Protection	Configuration	headers fields. Using the HTTP body or headers ensures sensitive data is encrypted properly while in transit.
System and Communications Protection	URL Parameters	URL parameters are not encrypted and should not be used to send sensitive data.
System and Communications Protection	Personal Information Collection	The application must provide clear instructions to the user on what (if any) personal information is collected and how it is used.
System and Communications Protection	Personal Information Collection	The application should provide an option the user can select to opt-in for personal data use.
System and Communications Protection	Sensitive Data Classification	An inventory list must be created and maintained for all sensitive data used and processed by the application to ensure it is protected properly through the use of encryption and retained according to industry regulations (if applicable).
System and Communications Protection	Cryptographic Key Protection	The application must use a Humana approved standard for protecting cryptographic keys.
System and Communications Protection	Cryptographic Key Protection	The application must follow the Humana standard for managing the lifecycle of cryptographic keys.
System and Communications Protection	Session Identifier Protection	The application must generate session tokens with at least 64-bits of entropy to avoid duplicate or predictive session identifiers. Additionally, session tokens must be generated using an approved Cryptographically Secure Pseudo Random Number Generator.
System and Information Integrity	URL Parameters	Sensitive data, such as secrets, API keys, and user session tokens must be sent in either the HTTP body or headers to prevent exposure. Data sent as part of the URL is not encrypted and may be intercepted by an attacker. Example of insecure method for sending sensitive data: https://mywebsite.com?apiKey=9870435
System and Information Integrity	URL Parameters	Use an allow list of acceptable Content-Type values and verify for each request.
System and Information Integrity	URL Parameters	The following HTTP methods should be disabled. Use of these methods can allow users to modify/delete files on the server or steal user credentials: PUT, HEAD, DELETE, CONNECT, TRACE If one of the above methods must be used limit use to an allow list of users and specific endpoints (e.g., /Specific/URL).
System and Information Integrity	URL Parameters	The application must validate JSON schema prior to use to prevent command injection. For more information please see: https://json-schema.org/
System and	Uploaded File	The file type, like all user input, must not be trusted by the

Control Family	Control Name	Control Requirement
Information Integrity	Protection	application. An allow list of approved file extensions must be defined and verified by the application. The file must be inspected to ensure the actual type matches the user-supplied file extension (prevent renaming an executable file with a .txt extension). Lastly, the Content-Type HTTP header must not be used to validate the file type as the value can be manipulated.
System and Information Integrity	Uploaded File Protection	Uploaded file names must be either renamed to a value set by the application or validated using a regex configured for acceptable values. Using a regex or application generated name avoid introducing risk to the system caused by use of restricted filenames and special or non-acceptable characters.
System and Information Integrity	Uploaded File Protection	File size limits must be enforced to protect file storage capacity. Additionally, for compressed files the uncompressed file size limit should be evaluated to prevent zip bombs, also known as billion laughs attack that can fill storage which hinders and damages the host's ability to function properly.
System and Information Integrity	Uploaded File Protection	Assign role based access controls to limit permissions to only users required to upload files.
System and Information Integrity	Uploaded File Protection	Do not store uploaded files in the installation directory of the web host application. Instead, uploaded files must be stored on a separate host or outside the web host's installation directory. Storing files outside this directory will prevent execution of malicious files can be used to modify the application.
System and Information Integrity	Uploaded File Protection	The application must use anti-virus software or a sandbox to execute and validate the uploaded file is free of malicious content. This ensures every uploaded file is evaluated for malicious content.
System and Information Integrity	User Supplied Input	The application must never insert input directly into any type of command. Instead, encode, escape, or sanitize input prior to use.
System and Information Integrity	User Supplied Input	An API's interpreter can be vulnerable to injection attacks. Instead, use an implicit input parameter validation by using strong types like numbers, booleans, dates, times or fixed data ranges in API parameters.
System and Information Integrity	User Supplied Input	The application must validate input prior to use, rejecting invalid input. If invalid input is not rejected properly the application may enter an unknown state which results in a denial of service or revealing sensitive information about the system helpful to an attacker.
System and Information Integrity	User Supplied Input	Input must be denied by default in order to reduce the attack surface. Input must only be allowed by the application in a defined list of locations.
System and Information	User Supplied Input	All input data must be validated for length, allowed characters using an allow list and patterns (regex) where applicable.

Control Family	Control Name	Control Requirement
Integrity		
System and Information Integrity	User Supplied Input	Input validation errors should be logged to notify security personnel of attempted injection attacks. Additionally, input data must be sanitized before writing to the log to prevent injection attacks on the logging solution.
System and Information Integrity	File & URL Path	The application cannot trust user input when constructing a file or URL path. The requested resource should not be stored in body of the HTTP request to avoid introducing file traversal vulnerabilities.
System and Information Integrity	HTTP Request Smuggling	The application must use specific logic for handling requests that contain both the Content-Length and Transfer-Encoding headers.
System and Information Integrity	Changeable Session Token	The application must by designed to allow changeable session identifiers. Using changeable session identifiers ensure each user session receives a new, randomly generated value that if compromised has a finite window of use by an attacker.
System and Information Integrity	HTTP Parameter Pollution Attacks	URL parameters must be verified server-side by validating all user input and accepting only one instance of a parameter. If multiple instances of a parameter are included then it should be dropped. HTTP Parameter Pollution attacks occur when an attacker supplies multiple HTTP parameters with the same name that may cause an application to interpret values in unanticipated ways. By exploiting these effects, an attacker may be able to bypass input validation, trigger application errors or modify internal variables values.
System and Services Acquisition	Third Party Components	Use only Humana verified and approved third-party components, hosted by the Humana standard solution. This ensures components are free from malicious content and vulnerabilities. Examples of third-party components include open source libraries, web components, and NodeJS modules.
Access Control	Kubernetes RBAC	Role-based access controls, with granular permissions, must be created to associate groups, actions (verbs), and resources (pod, services, nodes). Example: Group Name: pod-reader Verbs: get, watch, list
Access Control	Kubernetes RBAC	Access to Kubelets, the primary "node agent" that runs on each node, must deny un-authenticated access by default.
Access Control	Kubernetes RBAC	Granular access control permissions for all administrative operations in a service mesh (e.g., policy specifications, configuration parameters for service resiliency parameters, canary deployments, retries, etc.) must be specifiable at the level of all services in a namespace, a named service wtihin a namespace, etc.
Access Control	Kubernetes Network Access	The Kubernetes cluster must deny access to these ports by default. Denying access to these ports would prevent an attacker from quickly identifying the service and reduce the attack

Control Family	Control Name	Control Requirement
		surface of the cluster.
Access Control	Kubernetes Clusters	The Kubernetes dashboard must deny access from internal pods. If configured incorrectly the Kubernetes dashboard can grant access to a number of high-value permissions. Since internal pods reside on the edge an attacker can could gain access to the Kubernetes dashboard if access is allowed.
Access Control	Kubernetes Clusters	The service mesh service proxy must specify an allow list of protocols and ports which it can accept traffic for an associated service. Defining which protocols and ports are allowed at the service proxy level prevent unauthorized enumeration and access to containers from lateral microservice networks.
Access Control	Kubernetes Clusters	"Containers used in the Kubernetes cluster must be configured to deny traffic by default. This prevents un-authorized lateral network movement to another pod as well as outbound traffic. Example scenarios: 1. A container that has been compromised with remote access can allow an attacker to move laterally to another pod that is not available from external traffic. 2. An attacker can exfiltrate sensitive data stored on a compromised container because outbound traffic is allowed."
Access Control	Kubernetes Clusters	The microservices services mesh must also deny east-west traffic by default. Denying traffic by default prevents un-authorized lateral network movement other Kubernetes pods.
Access Control	Kubernetes Clusters	The Kubernetes cluster must be configured to deny direct SSH access to Kubernetes nodes. Instead, use "kubectl exec" for access. Using "kubectl exec" instead of direct SSH access provides access to the container environment and prevents access to host resources.
Access Control	Microservices Configuraton	The service mesh must enforce an allow list for outbound connections that microservices are authorized to communicate with, denying connectivity if not defined. Using a service mesh to mediate all egress traffic allows control and management over external services and how containers interact with them.
Access Control	Kubernetes Dashboard	The Kubernetes dashboard must use a low privileged non-interactive account. Using a low privileged account prevents access to high-value functions in the Kubernetes dashboard if the account is compromised.
Access Control	Kubernetes Dashboard	If the Kubernetes Dashboard must be accessible via public internet, then strong multi-factor authentication must be enabled. Strong multi-factor authentication controls include hard token or soft token. Do not use email or SMS.
Audit and Accountability	Microservices Configuraton	Containers must write log files to their local file system. Writing logs to the file system of a microservice preserves log data when combined with the other security requirements in this section.
Audit and Accountability	Microservices Configuraton	"Microservices must use a separate, decoupled, and dedicated logging agent to collect log data from the microservice and send

Control Family	Control Name	Control Requirement
		it to the logging message broker. Using a dedicated and decoupled logging agent mitigates the threat of data loss due to a failure in the log service (attack via denial of service, service interruption, etc.) Additionally, this prevents data loss as a result of logging agent failure by allowing the microservice to continue to write data to the file system. "
Audit and Accountability	Microservices Configuraton	The logging agent and message broker must use mutual authentication (mTLS) to authenticate and encrypt log messages. Using mTLS prevents microservice spoofing, logging/transport system spoofing, network traffic injection, and sniffing network traffic
Audit and Accountability	Microservices Configuraton	The SIEM solution must subscribe to the message broker to receive logs from microservices. Subscribing to log message broker prevents data loss if the logging service is offline or under attack.
Audit and Accountability	Kubernetes Metrics	Baseline activity must be developed to define "normal" behavior for microservices cluster. Examples of metrics to include in the baseline are averages of usage for CPU, memory, network bandwidth, etc. Establishing a baseline can aid personnel in early detection of security events.
Audit and Accountability	Kubernetes Metrics	Alerts must be generated and sent to the appropriate personnel if suspicious activity is detected. Automated alerts can assist with early detection and action of security events.
Identification and Authentication	API Gateway	The API gateway must integrate with the Humana standard solution for identity and access management. Using the API gateway's integration capabilities will allow for centralizing authentication policies of microservices.
Identification and Authentication	Microservices Authentication	All microservices-based authentication tokens must be digitally signed to provide assurances of authenticity and integrity. Digitally signing authentication tokens ensures messages cannot be manipulated in transit or forged by an attacker.
Identification and Authentication	Microservices Authentication	All microservices-based authentication tokens must be encrypted in transit using mTLS to prevent an attacker from intercepting the messages.
Identification and Authentication	Cloud Communication	Communication from clients may only begin after successful authentication to the API gateway.
Identification and Authentication	Cloud Communication	Communication from microservice-to-microservice may only occur after successful authentication via mutual TLS per transaction.
Configuration Management	Microservice Availability	The service mesh must safeguard availabilty of services by re-balancing load when limits are met, limiting the number of incoming requests to ensure availability of microservices, and avoiding unhealthy microservices.
Configuration	Container	Containers must use unprivileged accounts for running

Control Family	Control Name	Control Requirement
Management	Privilieges	application processes. Using unprivileged accounts prevent privilege escalation from a compromised microservice.
Configuration Management	Container Privilieges	Containers must be deployed in a read-only state to prevent changes to the file system. Denying write actions to the filesystem will prevent an attacker from uploading malicious files or making un-authorized changes to the microservices configuration.
Configuration Management	Container Privilieges	Use only Humana verified and approved container images. Using Humana approved container images ensure security configurations and vulnerability scans are performed prior to use. Using an unvetted container image may result in deploying a container that has been infected with malware or running an unpatched vulnerability.
Configuration Management	Container Privilieges	Containers must be sandboxed to isolate themselves from the host kernel. Use of sandboxing technologies prevent container breakout and kernel exploits.
Configuration Management	Container Privilieges	A proxy must be deployed to limit the trusted component to the proxy. Using a proxy between clients and microservices avoids the need to place the trust on the microservices. Additionally, this shields microservices from a client that submit large requests (e.g. denial of service attack).
System and Communications Protection	Cloud Certificates	Wildcard certificates must be used in as few locations as possible. Wildcard certificates allow validation of any address on a given domain and are high-value resources for attackers to steal. Limiting use of wildcard certificates reduces the likelihood of theft.
System and Communications Protection	Cloud Certificates	Certificate renewals must be automated to avoid service disruption and reduce errors in the renewal process introduced through manual steps.
System and Communications Protection	Cloud Certificates	Certificates issued by the service mesh for mTLS authentication must chain up to Humana's public key infrastructure. Using the existing public key infrastructure simplifies management of certificates by re-using the existing trust chain and revocation processes.
System and Communications Protection	Cloud Certificates	mTLS certificates must use a validity period that meet or exceeds Humana standards. Following Humana standards for certificate validity period ensure certificates are rotated on a regular basis and reduce risk if a private key is compromised.
System and Communications Protection	Cloud Certificates	The service mesh proxy must gracefully rotate its identity certificate upon renewal to prevent a disruption of services. Example: During certificate rotation a service mesh will gracefully end existing connections (established under the old certificate) and establish new connections using the newly issued certificate without disrupting services.
System and	Cloud Certificates	Identity certificates, used by microservices, should not include

Control Family	Control Name	Control Requirement
Communications Protection		the ability to sign other certificates. This prevents an attacker from issuing 'rogue' or unmanaged certificates that chain up to a trusted root if a certificate private key is stolen.
System and Communications Protection	Cloud Certificates	The service mesh should only issue an mTLS certificate to a microservice after the identity is verified. Ensuring the identity of a microservice prior to issuing a certificate will prevent unauthorized access to the service mesh.
System and Communications Protection	Cloud Certificates	Using DNS or a secure discovery service ensures that the server is the authorized location for the microservices and protects against network hijacking
System and Communications Protection	Microservices API Gateway	Access tokens must be translated at the API gateway to prevent leaking session tokens, used internally by microservices, back to clients.
System and Communications Protection	Microservices API Gateway	Access tokens, used internally by microservices, must never be passed back to clients. Preventing access tokens from being sent back to the client prevent an attacker from abusing credentials used by a microservice. Since each microservice carries it's own set of credentials, if leaked an attacker could gain unauthorized access to the microservices cluster.
System and Communications Protection	API Management	The microservices cluster must route all client-based traffic through an API gateway. Routing traffic through an API gateway allow for centralized enforcement of authorization downstream of all microservices and prevent having to manage individual authorization for each service.
System and Communications Protection	API Management	New microservices must route traffic through the same path as the existing microservices, through the API gateway. Leveraging the API gateway ensures authorization is centralized and prevent introducing separate authorization policies that may have gaps and allow anonymous access.
Identification and Authentication	API IAM Standards	The API should only use approved Humana standards for IAM.
Identification and Authentication	API Credential Data Storage	The API should access all data stores using unique credentials and multifactor authentication. All passwords must be managed using the Humana approved secrets key management for creating, storing, managing access, rotating and destroying secrets.
Access Control	API Password Policy	The API should adhere to password policy restrictions, to protect against brute force account and attack mechanisms. Following are password requirements to be met (at minimum): a) Enforce a minimum complexity - must be at least 8 characters - must have at least one special character - must contain at least one number - cannot contain User profile ID or name - cannot contain User profile ID or name backwards - cannot have three or more concurrent occurences of the same character - restrictions are applied to prevent extremely weak and easily guessable

Control Family	Control Name	Control Requirement
		passwords, inclusing exclusion list of commonly used passwords and password fragments b) Stores and transmits only cryptographically-protected passwords c) Requires password rotation (change) every 90 days (Humana Government Business: 60 days) d) Requires password not reused for 850 days
Access Control	API Anti-CSRF	The API should implement anti-CSRF mechanisms. If the API utilizes cookies, ensure they are protected from CSRF via the use of double submit cookie pattern or CSRF nonces. Additional implementation options include: * Verifying the origin header * Not disabling the same origin policy * Use SameSite cookies
Access Control	API Requests	If API keys are used, ensure they are implemented for every endpoint. While API keys are not generally considered secure (e.g., typically accessible to client), they can be beneficial for the following: * Block anonymous traffic. API keys identify an application's traffic for the API producer, in case the application developer needs to work with the API producer to debug an issue or show their application's usage * Controlling the number of API calls * Identifying usage patterns in the API's traffic * Filtering logs by API key
Access Control	API Requests	The API should return "429 Too Many Requests" HTTP response code when a DOS attack is detected or the request is rejected due to rate limiting.
Access Control	API HTTP Enforcement	The application must maintain an allow list of approved HTTP verbs per page and validate each request. Verify the API HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources. Failing to properly verify HTTP verbs minimizes protections against authentication or access control bypass attacks: https://web.archive.org/web/20081116154150/http://www.aspectsecurity.com/documents/Bypassing_VBAAC_with_HTTP_Verb_Tampering.pdf
Access Control	API HTTP Enforcement	The API should reject all requests with HTTP methods not found in the allow list with the HTTP response code "405 Method not allowed"
Access Control	API JWT Token Verification	The API should verify any JWT provided as an OAuth bearer token on each API request. If a request has an invalid JWT, the API should reject the request with a "403 Forbidden" error code. In the event that performance is a concern, the API may use caching technology such as REDIS to minimize direct calls to the DB
Access Control	API JWT Token Verification	The API should require that the JWT algorithm parameter (alg) in the JWT header match the type of JWT signature expected from the authentication provider. If the alg field differs from the expected value, (e.g., an API expects that AAD signs its JWTs using RS256, but finds a value of HS256), the API should reject

Control Family	Control Name	Control Requirement
		the request with a "403 Forbidden" error code
Access Control	API JWT Token Verification	The API should reject (with "403 Forbidden" error code) any attempt to authenticate with a JWT with the alg algorithm value set to "none".
Access Control	API JWT Token Verification	APIs should validate the "Not Before" (nbf) and "Expiration Time" (exp) fields to validate that they are before and after the current time, respectively. If an API receives a request that fails this validation check, the API must reject the request with a "403 Forbidden" error code
System and Communications Protection	API Secure Management	Ensure the API utilizes a secure management and Humana approved security configuration API management tool (e.g., IBM APIC/APIGEE). The management tool will provide an intake process to ensure APIs have been properly hardened.
System and Communications Protection	API Brute Force Prevention	Anti-automation controls such as CAPTCHA, lockout limits, rate limiting, etc. must be implented to guard user accounts from gaining unauthorized access through bute forcing automation attacks. It is recommended to implement a number of these techniques as a defense-in-depth approach, as no singular approach can guard against all types attacks.
System and Communications Protection	API TLS Policy	All APIs must utilize encryption at all times such as HTTPs and TLS 1.2 (1.3 preferred).
System and Communications Protection	API Error and Exception Prevention	The API should use a valid error handling framework for providing helpful messages to users, but not overly verbose information that is useful to an attacker.
System and Communications Protection	API Network Configuration	Ensure the API is secured behind the Humana approved network configuration using web/application gateway protection.
System and Communications Protection	API GUID Generation	Ensure the API generates any Globally Unique Identifiers (GUID) used via a cryptographically secure psuedo random number generator (CSPRNG) with a GUID v4 algorithm.
System and Information Integrity	API Sensitive Information Check	For POST/PUT requests, sensitive data should be transferred in the request body or request headers. Avoid exposing sensitive information (e.g., API keys, session tokens, etc.) in the URL, as this can be captured in web server logs. If sensitive information must be included in GET requests, ensure the content is sent in an HTTP header (and not in URL query strings).
System and Information Integrity	API Request Validation	All untrusted input must be validated for length, allowed characters using an allow list and patterns (regex) where applicable.
System and Information Integrity	API Request Validation	All parameterizations and variations of data stores and queries must be defined, validated, and sanitized.
System and	API Request	Verify that JSON schema validation is in place and verified

Control Family	Control Name	Control Requirement
Information Integrity	Validation	before accepting input.
System and Information Integrity	API Request Validation	Use a well known library and/or the frameworks built-in features (if available) to avoid introducing vulnerabilities into the API. Custom sanitizers may not account for all scenarios and leave the API vulnerable.
System and Information Integrity	API Request Validation	The API should define an appropriate request size limit and reject requests exceeding the limit with HTTP response status "413 Request Entity Too Large"
System and Information Integrity	API Request Validation	Ensure the API rejects all invalid API requests.
System and Information Integrity	API Request Validation	The API must validate input prior to use and reject invalid input. If invalid input is not rejected properly the API may enter an unknown state which results in a denial of service or revealing sensitive information about the system helpful to an attacker.
System and Information Integrity	API Status Codes	The API should respond to succesful actions with a 2xx code. This class of status codes indicates the action requested by the client was received, understood, and accepted For example: * 200 OK (response to a successful API action) * 201 Created (the request has been fulfilled and resource created) * 202 Accepted (The request has been accepted for processing, but processing is not yet complete)
System and Information Integrity	API Status Codes	The API should respond to requests with more than one possible response with a 3xx code. This class of status codes indicates the client must take additional action to complete the request. Many of these status codes are used in URL redirection. For example: * 301 Moved Permanently * 304 Not Modified * 307 Temporary Redirect
System and Information Integrity	API Status Codes	Verify that API requests containing unexpected or missing content types are rejected with appropriate headers. For example: * 400 Bad Request * 401 Unauthorized * 403 Forbidden * 404 Not Found * 405 Method Not Acceptable * 406 Unacceptable * 413 Payload too large * 415 Unsupported Media Type
System and Information Integrity	API Status Codes	The API should respond to 5xx status codes when the server has encountered an error or is otherwise incapable of performing the request. For example: * 500 Internal Server Error * 501 Not Implemented * 503 Service Unavailable
System and Information Integrity	API Content Type	The API should include the "Content-Type" header with the appropriate content type in requests for media sources. Furthermore, verify that the API explicitly checks the incoming Content-Type to be the expected value, such as application/xml or application/json.
System and	API Content Type	The API should prevent sensitive data from being stored/cached

Control Family	Control Name	Control Requirement
Information Integrity		by enforcing the "Cache-Control" HTTP request headers to prevent caching
System and Information Integrity	API Content Type	The API should use the "Content-Security-Policy" header as a layer of security for helping detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. Note - if the API does not handle HTML to be rendered on the client, the header may provide little or no security benefits.
System and Information Integrity	API Content Type	The API should use the "Strict-Transport-Security" header to prevent data from being sent unencrypted via HTTP.
System and Information Integrity	API Content Type	Use the "X-Frame-Options" header with a value of DENY, SAMEORIGIN, ALLOW-FROM-Uri. Note - if the API does not include content to be loaded in a frame or iFrame content the header may provide little or no security benefits. https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html#x-frame-options-header-types"
System and Information Integrity	API Content Type	Use the "X-Content-Type-Options" header with a value of "nosniff" to prevent MIME Sniffing, which allows users to transform non-executable content into executable content.
System and Information Integrity	API Content Type	An attacker can redirect a client to load resources from a malicious domain if the "Access-Control-Allow-Origin" header is set to an asterisk (*). By delivering appropriate CORS Headers, the API signals to the browser which domains, AKA origins, are allowed to make JavaScript calls. Note: * Disable CORS headers if cross-domain calls are not supported/expected. * Be as specific as possible and as general as necessary when setting the origins of cross-domain calls.
System and Information Integrity	API Schema Definitions	Throughly document the API formats and schema definitions. Traditional documentation can be useful for reviews by a less technical audience, but such forms of documentation are not easily maintained. The API schema definition formats are designed for quick generation of documentation as part of API design and mocking that is also reusable for testing, integration, publishing, and operations. APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important
System and Information Integrity	API Schema Definitions	Humana shall adopt versioning of API releases using a two part versioning scheme: {major version}.{minor version}. The major version number is incremented when a backwards-incompatible change occurs. The URLs used in the Humana API should reflect this by incorporating the major version.
System and Information Integrity	API Labeling	Ensure the API is properly labeled and tagged as per Humana approved standards such that identity information is known for the Humana defined classification level, platform environment

Control Family	Control Name	Control Requirement
		and type of API (e.g., internal, external and version number).
Audit and Accountability	API Data Approval	The API must be reviewed and approved by a Data Steward and/or data security assessment/analysis.
Audit and Accountability	API Request Logging	Ensure logging of API requests adheres to Humana logging best practices.
Security Assessment and Authorization	API Vulnerability Management	Ensure the API undergoes vulnerability assessment and/or vulnerability management scanning.
Security Assessment and Authorization	API Vulnerability Management	Ensure the API code is periodically analyzed with static analysis tools upon code commits
Security Assessment and Authorization	API Vulnerability Management	Ensure the API is tested with fuzzers and DAST solutions
Security Assessment and Authorization	API Vulnerability Management	Ensure the API is pentested periodically

Appendix

A. Scoping Questionnaire Responses

S.No	Design Category	Question	Response	Notes
1	General	List the Operating System(s), Application Server(s), Database Server(s), etc. with version numbers	Kubernetes version: 1.31.5,Operating system: Ubuntu Linux,Virtual machine size:Standard D4ds v5 (4 vcpus, 16 GiB memory)	
		How many and what type of major user roles exist within the application, e.g. Standard User, Super User, Manager, Admin, etc.?	Azure Kubernetes Service Cluster User Role,Azure Kubernetes Service RBAC Reader,Reader,Contributor	
		What are the main technologies and SDK's used for the application? (Please include web browser, database platforms, messaging, API's, 3rd party components, 3rd party SDKs). For any 3rd party components, please indicate the version implemented in the app.	Java 17, Springboot 3.4.X, Kafka, Postgres, APIGEE, Kubernates 1.31.5,	
		What programming language(s) and version is the application developed in?	Java 17	
		Approximately how many lines of code (LOC) are in the application?	10K - 50K	
2	Content & Features	What is the approximate total number of web services or landing pages?	5	
		What percentage of the information contains dynamic content?	10	
		Does the application services include features or functionality that should be noted for scoping purposes?	Yes	
3	Application Services Architecture and Environment	Briefly describe the application service architecture and environment (e.g. Operating System, HTTP Server, Web Programming Languages)	"It is a microservices-based application built with Spring Boot, where Kafka is used as the messaging service for communication between the services. "	

S.No	Design Category	Question	Response	Notes
		Has an application architecture review or a security architecture review been performed? Is the review documentation available and can be shared? Who was the person from EIP that performed the assessment?	Application architecture review was done with Alex	
		Does the application utilize a load balancer?	Yes	
		Is the application hosted by a third party or hosting provider?	Yes	
		What infrastructure security controls are in place that provide a level of defense to the application, e.g. web application firewalls, key management, proxy servers, token manager, SSO, etc.?	Web application firewalls, proxy servers and token manager	
		What mechanisms are used for protecting data at rest and during transmission (state the type of cryptographic processes and algorithms used)?	NA	
4	Communication Protocols	Does the application utilize Web Services (HTTP protocol in conjunction with XML, SOAP, WSDL and UDDI technologies)?	Yes	
		Does the application utilize Asynchronous JavaScript and XML (AJAX)?	No	
		Does the application provide API services or use API Services (please provide list of the services and mark as either service or consumer)?	Yes	
5	Testing Environment	What type of testing is performed against the application and services, e.g. Static Analysis, Dynamic Analysis, Pen testing, Fuzzing, etc.?	Static Analysis, Dynamic Analysis, Load testing	
		What are the frequencies of each testing methodology?	yes release wise	
		Are security controls in the testing environment equivalent to the production environment?	yes UAT	

B. ASA Detailed Assessment

S.No	Design Category	Question	Response	Notes
1	Access Control	1.1 - Is access to privileged functions restricted by the enforcement of role-based access controls or attribute-based access controls? Please answer RBAC/ABAC/Neither and provide details.	Fully Implemented	Fully Implemented, Using Azure AD for the authentication
		1.2 - Does the access control system require users with access to security-related information to use non-privileged accounts when performing non-security functions?	Fully Implemented	Fully Implemented, Using Azure AD for the authentication
		1.3 - Is the access control policy enforced over user accounts with permissions to the application and application resources (e.g., are there no gaps across the application where access control policies are not/cannot be applied)?	Fully Implemented	Fully Implemented, Using Azure AD for the authentication
		1.4 Is access to data repositories restricted to the least number of users required?	Fully Implemented	
		1.5 Does the RBAC/ABAC system allow access for users to be revoked, reduced, or altogether disabled?	Not Applicable	
		2.1 - Are Humana approved automated mechanisms used to manage system/service/privileged/unprivileged accounts?	Fully Implemented	
		2.2 - Is the removal/disabling of temporary and emergency accounts automated to execute after a specified period of time?	Not Applicable	
		2.3 - Are accounts evaluated for the following conditions and automatically disabled after a specified period of time? - Expired -No longer in use -Inactive after a defined period of time -User leaves the organization	Not Applicable	
		2.4 - Does an automated process exists for all application account events: creation, modification, enabling, disabling, and removal of permissions?	Fully Implemented	
		2.5 - Does a process exist to periodically review and update account privileges as account permissions change over time?	Fully Implemented	
		2.6 Are non-interactive service accounts used under conditions	Not	

S.No	Design Category	Question	Response	Notes
		that align with Humana standards?	Applicable	
		3.1 - Is directory browsing limited to only directories where it is required?	Not Applicable	
		3.2 - Does the application's directory browsing features prevent the discovery of hidden files such as Thumbs.db, .DS_Store, .git or .svn folders?	Not Applicable	
		4.1 - Does the application deny access to authenticated pages by default?	Not Applicable	
		4.3 Are permissions granted only when explicitly requested and after review?	Not Applicable	
		5.1 - Does the application verify permissions on all user-supplied requests to objects?	Not Applicable	
		6.1 - Does the user have to re-authenticate via a password or another method (i.e., 2nd token) before performing a high-risk transaction/privileged action?	Not Applicable	
		7.1 - Does the application implement any of the following anti-CSRF measures? -Use a framework built-in anti-CSRF protection -For stateful software use a synchronizer token. -Double submit cookies for stateful software. -Implement at least one mitigation from this article on defense in depth mitigations: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#defense-in-depth-techniques -Consider SameSite Cookie Attribute for session cookies but be careful to NOT set a cookie specifically for a domain as that would introduce a security vulnerability that all subdomains of that domain share the cookie. This is particularly an issue when a subdomain has a CNAME to domains not in your control. -Consider implementing user interaction based protections for highly sensitive operations. -Consider the use of custom request headers. -Verify the origin with standard headers.	Not Applicable	
		73.1 - Are role-based access controls defined using a Humana approved solution, to associate users with groups, verbs with resources, and permissions of microservices scoped to namespace or cluster?	Not Applicable	
		73.2 - Is un-authenticated access to Kubelets disabled?	Fully Implemented	Disabled by default in Kubernetes
		73.3 - Are granular access control permissions defined for the service mesh and scoped to namespace, named service, etc.?	Fully Implemented	Registered our Microservices in service mesh

S.No	Design Category	Question	Response	Notes
		74.1 - Are ports for the following Kubernetes services protected with strict network access controls? - Kubernetes API Server, TCP 6443 - etcd server client API, TCP 2379-2380 - Kubelet API, TCP 10250 - kube-scheduler, TCP 10251 - kube-controller-manager, TCP 10252 - Read-Only Kubelet API, TCP 10255 - NodePort Services, TCP 30000-32767	Fully Implemented	
		74.2 - Are network access controls enforced to deny access to the Kubernetes dashboard from internal pods?	Fully Implemented	
		74.3 - Does the service mesh service proxy use an allow list for accepting specific protocols/ports and deny by default?	Fully Implemented	
		74.4 - Do microservices use a default configuration to deny all east-west and north-south traffic by default?	Not Applicable	
		74.5 - Does the microservices service mesh proxy deny by default traffic for east-west traffic?	Not Applicable	
		74.6- Is direct SSH access to microservices in the Kubernetes cluster denied?	Fully Implemented	
		74.7 Is the service mesh configured to mediate all egress traffic through an egress proxy?	Fully Implemented	
		75.1 - Is the service account, used for the Kubernetes dashboard, a non-privileged user?	Fully Implemented	
		75.2 - If the Kubernetes dashboard is accessible from public internet, are strong multi-factor authentication controls enforced?	Not Applicable	
		90.1 - Does the API adhere to password policy restrictions, to protect against brute force account and attack mechanisms?	Not Applicable	
		91.1 - Does the API implement anti-Cross Site Request Forgery (CSRF) mechanisms?	Fully Implemented	
		92.1 - Are the API keys required for every request to the endpoint?	Fully Implemented	Because of Apigee
		92.2 - Does the API rate limit requests?	Fully Implemented	Through Apigee
		93.1 - Does the API implement and enforce an allow list of accepted HTTP methods (e.g., GET, POST, PUT)?	Fully Implemented	Yes, we have specified endpoints
		93.2 - Does the API reject all requests with HTTP methods not found in the allow list, and return the HTTP response code "405 Method not allowed?"	Fully Implemented	
		94.1 - Does the API verify the JWT on each API request?	Not Applicable	
		94.2 - Does the API check if the algorithm parameter "alg" in the JWT header matches the type of JWT signature expected	Not Applicable	

S.No	Design Category	Question	Response	Notes
		from the authentication provider?		
		94.3 - Does the API reject any attempts at authenticating with a JWT token with the "alg" algorithm value to "none"?	Not Applicable	
		94.4 - Does the API validate the "Not Before" (nbf) and "Expiration Time"	Fully Implemented	
2	Audit and Accountability	9.1 Do the application logs contain (at minimum) the following information for both application and security events: - A time stamp using NTP/SNTP - The severity level - An indication if this was a security event - The identity of the user that caused the event - The source IP address of the request - Whether the event succeeded or failed - A description of the event - The screen or page the user was accessing during the event	Fully Implemented	
		9.2 - Does application define a maximum limit to the log file size (e.g., prevent service disruption from a system drive that is out of free space)?	Fully Implemented	ARGO CD pipeline, From DB log file have limit set.
		9.3 - Does the application prevent the logging of sensitive data? Sensitive data can include passwords/secrets in clear text, PII, PHI, or PCI data.	Fully Implemented	PII and PHI data stores in our DB but not in the logs
		9.4 - Does the application perform output encoding when writing log data to a UI page?	Not Applicable	
		9.5 - Does the application restrict log file access to the least number of users as needed?	Not Applicable	
		9.6 - Are archived log files protected from tampering or destruction?	Fully Implemented	ARGO CD Pipeline
		9.7 - Are event logs stored in an approved Security Information and Event Management (SIEM) tool? (Splunk is recommended by Humana EIP)	Fully Implemented	
		9.8 - Are application logs transmitted to the SIEM solution encrypted in transit?	Fully Implemented	Splunk
		10.1 - Is a "last resort" error handler available across the entire application to catch all unhandled exceptions?	Fully Implemented	
		11.1 - Are logging controls enforced on the server-side?	Fully Implemented	
		11.2 - Can logging be disabled or configured by the user on client-side of the application?	Fully Implemented	
		76.1 - Do microservices write logs to their local file system?	Fully Implemented	
		76.2 Is a dedicated logging agent, decoupled from the	Not	

S.No	Design Category	Question	Response	Notes
		microservice, used to collect log data from the microservice and then send it to a message broker service?	Applicable	
		76.3 - Does the logging agent and message broker require mutual authentication (mTLS) before exchanging any logs?	Not Applicable	
		76.4 - Does the SIEM solution subscribe to the message broker service to receive logs from microservices?	Fully Implemented	Splunk and Dynatrace
		77.1 - Is a metrics baseline established for the Kubernetes cluster? A baseline establishes what "normal" behavior (metrics) look like for a Kubernetes cluster.	Fully Implemented	
		77.2 - Are alerts generated when suspicious activity occurs (i.e., deviations from baseline monitoring) ?	Not Applicable	
		108.1 - Does the API adhere to Humana API logging best practices (e.g., consistent formats, clock and time zones synchronization, log forwarding to SIEM)	Fully Implemented	
		107.1 - Has (or will) the API be reviewed and approved by a Data Steward and/or data security assessment/analysis?	Fully Implemented	
3	Configuration Management	14.1 - Do HTTP responses from the application include a "Content-Type" header with a safe character set to validate client content for the type?	Fully Implemented	
		14.2 - Do application responses contain a "Content-Disposition" header with the appropriate filename for the type of content?	Not Applicable	
		14.3 - Does the application include the HTTP header Cross Origin Resource Sharing (CORS) to load resources from a defined list of allowed domains?	Not Applicable	
		14.4 - Does the application use the HTTP header, Strict-Transport-Security (HTTP Strict Transport Security/HSTS)?	Fully Implemented	
		14.5 - Does the application use the Content-Security-Policy header?	Not Applicable	
		14.6 - Does the application enforce the X-Frame-Options HTTP response header to prevent Clickjacking attacks?	Not Applicable	
		14.7 - Does the application use the X-Content-Type-Options: nosniff header?	Not Applicable	
		15.1 Are web or application server and application framework debug modes disabled in production?	Fully Implemented	
		16.1 - Does the application sanitize third-party components utilized in the application such as third-party libraries?	Not Applicable	
		16.2 - Is an inventory catalog maintained for all third-party libraries in use?	Not Applicable	
		17.1 Does the application use an allowed list of approved HTTP verbs per business need (for example: GET, POST,	Not Applicable	

S.No	Design Category	Question	Response	Notes
		PUT, PATCH and DELETE are the options)?		
		18.1 - Does the application use custom error messages with generic content when unexpected error occur?	Fully Implemented	
		19.1 - Are HTTP headers and HTTP responses purged of unnecessary information about the underlying host and components?	Not Applicable	
		20.1 - Does the application use an allow list of file types that may be served by the web application?	Not Applicable	
		21.1 Is an allow list used by the application for resources or systems to which the server can send requests or from which it can load data/files?	Not Applicable	
		80.1 - Does the service mesh proactively adjust traffic direction and load based on availability of microservices?	Fully Implemented	
		81.1 - Do microservices use unprivileged users for running application processes?	Not Applicable	
		81.2 - Do containers use a read-only state configuration for both root file system and mounted volumes?	Not Applicable	
		81.3 - Are only Humana approved container images used?	Fully Implemented	
		81.4 - Is a container runtime sandboxing technology used?	Not Applicable	
		81.5 - Is trust between clients and microservices limited to a proxy?	Implemented - Inherited	
4	Identification and Authentication	22.1 - Does the application utilize Humana approved Identity and Access Management (IAM) solutions for managing authentication?	Fully Implemented	
		22.2 - If shared accounts are supported, are users required to authenticate via their user account prior to gaining access to the shared account?	Not Applicable	
		22.3 - Does the application enforce multi-factor authentication (MFA) for privileged and non-privileged accounts?	Not Applicable	
		22.4 - Does the application accept only Humana-approved MFA solutions?	Not Applicable	
		23.1 - Is the application designed to deny access if user authentication controls fail or enter an unknown state?	Fully Implemented	Fully Implemented, Using Azure AD for the authentication
		24.1 - Do the application's password complexity requirements adhere to Humana Standards?	Not Applicable	
		24.2 - Are user passwords compared against a common or	Not	

S.No	Design Category	Question	Response	Notes
		breached password list during registration, login, and password reset events?	Applicable	
		24.3 - Is password expiration enforced and require users to change their password on a regular basis?	Not Applicable	
		24.4 - Does the application require MFA to protect user accounts from a password breach?	Not Applicable	
		24.5 - Do non-interactive accounts (service account, application account) use secrets (passwords, API keys, etc.) that rotate on a regular basis according to Humana standards?	Fully Implemented	
		24.6 - Does the application provide feedback to the user if the password exceeds the maximum character available in the field?	Not Applicable	
		25.1 - Is communication between all application components, including APIs and data layers, authenticated?	Fully Implemented	
		25.2 - Are all connections to and from external systems involving sensitive information authenticated?	Fully Implemented	
		26.1 - Does a process exist to recover a user account if a MFA device is lost?	Not Applicable	
		26.2 - Does the application require users to verify their identity through one of the following techniques if the MFA device is lost or unusable? -Providing the user with a number of single-use recovery codes when they first setup MFA. -Requiring the user to setup multiple types of MFA (such as a digital certificate, OTP core and phone number for SMS), so that they are unlikely to lose access to all of them at once. -Posting a one-use recovery code (or new hardware token) to the user. -Requiring the user contact the support team and having a rigorous process in place to verify their identity. -Requiring another trusted user to vouch for them.	Not Applicable	
		27.1 - Does the application enforce MFA?	Not Applicable	
		27.2 - Does the application enforce account lockout after a defined number of unsuccessful login attempts?	Not Applicable	
		27.3 - Are anti-automation protective technologies used as additional layers of security from brute forcing accounts? Anti-automation technologies include: - CAPTCHA - IP Restrictions - Rate Limiting / Time Delays - Geolocation	Not Applicable	
		27.4 - Does the application monitor and actively protect user accounts from suspicious login activity?	Not Applicable	
		28.1 - Does the application allow a user to change their password at will?	Not Applicable	
		29.1 - A user attempts a login. The page returns the message	Not	

S.No	Design Category	Question	Response	Notes
		"Password is invalid." Does the application use generic error messages during failed login and password reset events?	Applicable	
		29.2 - Is the HTTP response time the same between a successful and unsuccessful login attempt?	Not Applicable	
		30.1 - Does a logged in user have the option to logout from any screen/web page of the application?	Not Applicable	
		31.1 - Does the "forgot password" process use a OTP, soft token, or send an email with a URL containing a time-limited activation token?	Not Applicable	
		31.2 - Does the application use a separate channel(s) to communicate the method to reset password such as email or SMS text?	Not Applicable	
		31.3 - Do URL tokens for password reset links meet the following requirements? -Randomly generated using a Cryptographically Secure Pseudo Random Number Generator -At least 64-bits in length. -Stored securely until expiration or use -Single use and expire after an appropriate period.	Not Applicable	
		31.4 - Does the account prevent changes to an account until a valid token is presented during a forgot password event?	Not Applicable	
		32.1 - Does the application utilize built-in session management controls provided by the supporting framework? Framework examples include PHPSESSID (PHP), JSESSIONID (J2EE), ASP.NET_SessionId (ASP .NET)	Not Applicable	
		33.1 - If SMS and/or email are available for two-factor authentication, are users made aware of the potential risks of either (e.g., email address compromise allows for full access to account, SIM Swap grants attacker access to all phone calls and SMS messages)?	Not Applicable	
		34.1 - Are passwords protected using an auto-salting hashing algorithm such as one of the following algorithms? Argon2id bcrypt PBKDF2 AES2 for z/OS mainframe systems with ACF2 installed.	Not Applicable	
		34.2 - If an auto-salting secure hashing algorithm is not used, are salts generated using a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) and a minimum length of 32-bits?	Not Applicable	
		36.1 - Are the same security controls that are applied to the login page, also applied to the change password page?	Not Applicable	
		37.1 - Does the application change the session identifier if a user's privilege or role changes (e.g., switching from an anonymous user to authenticated)?	Not Applicable	
		38.1 - Are secrets (e.g., passwords, internal secrets, and API	Fully	

S.No	Design Category	Question	Response	Notes
		keys) not hardcoded in source code and instead persisted in a Human approved Vault?	Implemented	
		39.1 - Are passwords checked against a breached password list or service?	Not Applicable	
		78.1 - Are all microservices authentication tokens digitally signed? Note: Microservice/api that uses APIC and APIGEE generate tokens are digitally signed.	Fully Implemented	
		40.1 - Is a password strength meter presented to users during registration?	Not Applicable	
		78.2 - Are all microservices authentication tokens encrypted in transit?	Fully Implemented	
		79.1 - Does communication from client to the API gateway only occur after successful authentication?	Fully Implemented	
		79.2 - Does communication from microservice-to-microservice only occur after successful authentication?	Fully Implemented	
		77.3 - Does the microservices API gateway integrate with the Humana approved identity and authentication management solution?	Fully Implemented	
		88.1 - Does the API use the Humana approved standards for identity and access management (IAM)?	Fully Implemented	
		89.1 - Does the API access all data stores using unique credentials and multifactor authentication?	Fully Implemented	MFA is not Applicable
5	Security Assessment and Authorization	42.1 - Has a threat model of the application been completed?	Not Applicable	
		42.2 - Are all business logic steps verified by the application such as during password change when password reset token is verified?	Not Applicable	
		43.1 - Does the application have configurable alerting when automated attacks or unusual activity are detected?	Not Applicable	
		44.1 - Is the application tested for race conditions or "Time of Check to Time of Use" (TOCTOU) on application code and business logic?	Not Applicable	
		109.1 - Does the API undergo vulnerability assessment and/or vulnerability management scanning?	Fully Implemented	
		109.2 - Is the API code periodically analyzed with static analysis tools (e.g., code quality checkers, SAST) upon code commits?	Fully Implemented	
		109.3 - Is the API tested with fuzzers and DAST solutions	Fully Implemented	We have DAST Solutions
		109.4 - Is the API pentested periodically?	Fully	Sonar Qube,

S.No	Design Category	Question	Response	Notes
			Implemented	Checkmark security tools to identify any vulnerabilities
6	System and Communications Protection	45.1 - Is TLS 1.2+ and Humana approved encryption algorithms used to encrypt sensitive data in transit?	Fully Implemented	
		45.2 - Is sensitive data encrypted at rest using Humana approved encryption algorithms?	Fully Implemented	All requests are routed through Apigee
		46.1 - Is a list defined for all required secrets and storage locations (CyberArk, Hashicorp Vault) that will be used by the application?	Fully Implemented	
		47.1 - Does the application use custom cryptographic algorithms in lieu of algorithms in-built within the platform/framework in which the application is developed?	Implemented - Inherited	Via Apigee
		47.2 - Do symmetric operations use authenticated modes of encryption?	Fully Implemented	Via Apigee
		47.3 - Are cryptographic components properly configured according to security best practices? -Initialization vectors to be generated randomly and unique per each symmetric encryption transaction -Symmetric encryption minimum key size of 256 bits -Minimum hashing algorithm of SHA-2 -For asymmetric keys use minimum length of 2048-bit for RSA and 512 for Elliptic Curves	Not Applicable	Via Apigee
		47.4 - Does the application use a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) for generating random numbers (e.g., session tokens, initialization vectors, etc.)?	Not Applicable	Via Apigee
		48.1 - Does the application store sensitive data on the client or within the client's browser?	Not Applicable	Internal API, not user facing. Not applicable
		48.2 - Is sensitive information stored in memory?	Implemented - Inherited	
		48.3 - Are session authenticators (e.g., session token) cleared from client storage once the session has been terminated?	Not Applicable	Internal API, not user facing. Not applicable
		48.4 Does the application store session tokens in the browser using only secure methods such as appropriately secured cookies or HTML 5 session storage?	Not Applicable	

S.No	Design Category	Question	Response	Notes
		50.1 - Does the application use technologies to prevent denial of service via the rapid transmission of an excessive number of requests per: - IP address - User - Hour or day	Not Applicable	
		50.2 - Does the application use defined resource allocation quota's to prevent excessive provisioning from an abnormal increase in traffic?	Not Applicable	
		51.1 - Does the application use code signing to verify the integrity and authenticity of binaries or code components including 3rd party libraries prior to use?	Not Applicable	
		52.1 - Does the application use the following HTTP header attributes to protect the user's session identifier from theft? - Secure -HttpOnly -SameSite -Max-Age	Not Applicable	
		52.2 - Does the application uses a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) to generate session identifiers of at least 128-bits (16 bytes) in length?	Not Applicable	Via Apigee
		52.3 - Are detection methods used by the application to detect anomalies in session ID's.	Not Applicable	
		52.4 - Does the application log information regarding changes in sessions? Note - the session IDs should not be logged.	Not Applicable	
		52.5 - Has a review been performed to determine if simultaneous user sessions are allowed?	Not Applicable	Internal API, not user facing. Not applicable
		53.1 - Are cookie path attributes set using precise paths for applications published under a single domain, but reside on the same host as other applications?	Not Applicable	
		54.1 - Is the application configured to automatically terminate or logoff a user session after a time of inactivity?	Not Applicable	
		54.2 - Are user sessions terminated by the application immediately if a user clicks the "logout" option?	Not Applicable	Internal API, not user facing. Not applicable
		54.3 - Are new session identifiers generated every time a user re-authenticates or performs a login after being logged out?	Not Applicable	Internal API, not user facing. Not applicable
		54.4 - Does the application require re-authentication before allowing any sensitive data access, transactions, or account modifications?	Not Applicable	Internal API, not user facing. Not applicable
		55.1 - Has an inventory list been created for all digital certificates in use by the application?	Not Implemented	DNS

S.No	Design Category	Question	Response	Notes
		55.2 Has the domain name of the application been compared to the Subject Alternative Name DNS value of the application's digital certificate for consistency?	Fully Implemented	DNS
		55.3 Has a process been implemented to automatically renew the digital certificate's prior to expiration?	Fully Implemented	DNS
		55.4 - Are all certificates installed correctly?	Fully Implemented	
		56.1 - Are caching features of autocomplete forms disabled for clients?	Not Applicable	
		57.1 Are random values used in critical features and transactions generated with a Cryptographically Secure Pseudo Random Number Generator?	Not Applicable	
		58.1 - Is Subresource Integrity (SRI) used to validate the integrity of application assets such as JavaScript libraries or web fonts?	Not Applicable	
		59.1 - Is all sensitive data sent to the server in the HTTP message body or headers?	Fully Implemented	
		59.2 - Are URL parameters free of sensitive data?	Fully Implemented	
		60.1 - Is a disclaimer provided to the user that clearly informs about collection and use of personal information?	Not Applicable	
		60.2 - Does the application by default opt-out of using personal data unless a user elects to share their information?	Not Applicable	
		61.1 - Has a review been completed to identify the classification of data that will be processed by the application? Data may be classified as public, internal, confidential, restricted.	Fully Implemented	
		62.1 - Is cryptographic key material stored in a Humana approved cryptographic management solution?	Not Applicable	
		62.2 - Does the application follow the Humana standard for managing the lifecycle of cryptographic keys (e.g. generated, distributed, revoked, and expired)?	Not Applicable	
		63.1 - Do session tokens possess at least 64 bits of entropy?	Not Applicable	
		100.1 - If the API uses one or more Globally Unique Identifiers (GUIDs), are they generated via a cryptographically secure psuedo random number generator (CSPRNG) with a GUID v4 algorithm?	Not Applicable	
		82.1 - Is wildcard certificate use limited to as few locations as possible?	Not Applicable	
		82.2 - Is an automated process/technology used to	Not	

S.No	Design Category	Question	Response	Notes
		automatically renew digital certificates prior to expiration?	Applicable	
		82.3 - Do certificates, issued by the service mesh, chain up to Humana owned public key infrastructure?	Not Applicable	
		82.4 - Is a validity period that meet or exceeds Humana standards used when issuing certificates from the service mesh?	Not Applicable	
		82.5 - Does the service mesh proxy gracefully rotate it's identity certificate upon renewal?	Fully Implemented	
		82.6 - Is the signing capability removed from mTLS certificates issued to microservices?	Fully Implemented	
		82.7 - Does the service mesh's control plane certificate management system first authenticate a microservice before issuing an mTLS certificate?	Fully Implemented	
		82.8 - Does the service mesh use DNS or a secure discovery service to associate service identity to the micorservice name when issuing certificates?	Fully Implemented	
		83.1 - Does the microservices API gateway manage access tokens between client and API gateway as well as API gateway to microservices?	Fully Implemented	
		83.2 - Do the microservices (or their controls) ensure that access tokens used internally are not exposed beyond the API gateway back to the user	Fully Implemented	
		84.1 - Does the client route traffic through an API gateway?	Fully Implemented	
		84.2 - Does new versions of a microservice use the same API management gateway as the current version?	Fully Implemented	
		85.1 - Are the communication channels between client and API gateway encrypted?	Not Applicable	
		85.2 - Are the communication channels between microservice-to-microservice encrypted?	Not Applicable	
		96.1 - Does the API utilize automated defensive technologies to protect against account brute forcing attacks?	Not Applicable	
		97.1 - Does the API adopt the recommended TLS versions per Humana policy (e.g., TLS v1.2+)?	Fully Implemented	
		98.1 - Does the API respond to errors and exceptions with generic error messages (e.g., error messages do not disclose system information, call stack data, etc.)?	Fully Implemented	
		99.1 - Is the API secured behind the Humana approved network configuration using web/application gateway protection?	Fully Implemented	
7	System and	64.1 Are all API URLs designed to not expose sensitive	Fully	

S.No	Design Category	Question	Response	Notes
	Information Integrity	information such as API keys, secrets, user session tokens, etc.?	Implemented	
		64.2 - Are requests containing invalid or missing content type properly rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type)?	Fully Implemented	
		64.3 - Are HTTP methods properly validated to prevent normal, unauthorized users from using DELETE or PUT on protected API requests or resources?	Not Applicable	
		64.4 - Is JSON schema validation implemented and verified prior to accepting input?	Fully Implemented	
		65.1 - Does the application validate the file type of the uploaded file for an allowed type prior to use (for example, file extensions and MIME type)? A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes. MIME types are defined and standardized in IETF's RFC 6838	Fully Implemented	
		65.2 - Does the application change the file name to a value generated by itself or use a regex to validate?	Fully Implemented	
		65.3 - Is a file size limit enforced by the application?	Fully Implemented	Enforced at Database for Log file
		65.4 - Does the application restrict file upload capabilities to a limited number of users?	Not Applicable	
		65.5 - Does the application store uploaded files outside the root directory where the web host application is installed?	Fully Implemented	
		65.6 - Are uploaded files scanned by an anti-virus solution or validated by a sandbox?	Not Applicable	
		66.1 - Does the application utilize built-in framework validation solutions or well-known libraries for sanitizing user input?	Not Applicable	
		66.2 - Does the application encode/escape all user supplied input prior to use with any commands?	Not Applicable	
		66.3 - Does the applications API use a parameter interface instead of an interpreter?	Fully Implemented	
		66.4 - Does the application remain stable when invalid inputs are received?	Fully Implemented	
		66.5 - Does the application deny allowing input by default?	Fully Implemented	
		66.6 - Does the application validate all input for length, allowed characters, and patterns on the server?	Fully Implemented	

S.No	Design Category	Question	Response	Notes
		66.7 - Are all input validation failures logged and properly sanitized prior to being logged?	Fully Implemented	
		67.1 - Does the application replace all user-supplied input with static or sanitized values for the construction of a file path or URL?	Not Applicable	
		69.1 - Does the application accept only one of the following: Content-Length or Transfer-Encoding headers?	Not Applicable	
		70.1 - Does the application use changeable session tokens rather than static API keys, secrets, passwords?	Fully Implemented	
		71.1 - Does the application parse input parameters and avoid accepting concatenated parameters, accepting only the first or last input?	Fully Implemented	
		106.1 - Is the API properly labeled and tagged as per Humana approved standards?	Fully Implemented	
		105.1 - Is the API's formats and schema definitions documented?	Fully Implemented	
		105.2 - Is versioning of API releases documented?	Fully Implemented	
		104.1 - Does the API include a "Content-Type" header with the appropriate content type in requests for media sources?	Fully Implemented	
		104.2 - Do the API responses contain a "Cache-Control" header with the appropriate value to prevent sensitive information from being cached?	Fully Implemented	
		104.3 - Does the API use the "Content-Security-Policy" header?	Fully Implemented	
		104.4 - Does the API use the HTTP header "Strict-Transport-Security" (HSTS)?	Fully Implemented	
		104.5 - Does the API use the "X-Frame-Options" header?	Not Applicable	
		104.6 - Does the API use the "X-Content-Type-Options" header?	Fully Implemented	
		104.7 - Does the API include the HTTP header Cross Origin Resource Sharing (CORS) "Access-Control-Allow-Origin" to load resources from a defined list of allowed domains?	Fully Implemented	
		103.1 - Does the API respond with 2xx codes for successful actions?	Fully Implemented	
		103.2 - Does the API respond with 3xx codes when the request has more than one possible response?	Fully Implemented	
		103.3 - Does the API respond with 4xx codes when the server is unable to process the request due to something perceived as a client error (e.g., invalid input)?	Fully Implemented	

S.No	Design Category	Question	Response	Notes
		103.4 - Does the API respond with 5xx codes when the server encounters an unexpected condition preventing it from fulfilling the request?	Fully Implemented	
		102.1 - Does the API validate untrusted input for its length, range, format, and type?	Fully Implemented	
		102.2 - Are all parameterizations and variations of data stores and queries defined, validated, and sanitized?	Fully Implemented	
		102.3 - If the API utilizes the JSON format, does the API validate and verify the JSON schema before accepting input?	Fully Implemented	
		102.4 - Does the API utilize available validation and sanitization libraries provided by the framework?	Fully Implemented	
		102.5 - Does the API define an appropriate request size limit?	Fully Implemented	Yes w.r.t Request payload
		102.6 - Are all invalid API requests rejected?	Fully Implemented	If an API receives improperly formatted JSON, XML, or parameters it returns bad request as well for nonexistent endpoints
		102.7 - Does the API remain stable when invalid and unexpected input is received?	Fully Implemented	APIs has strong input validation returns clear error messages (e.g., HTTP 400 for bad requests) rather than crashing.
		101.1 - Does the API handle sensitive information?	Fully Implemented	PII & PHI Data
8	System and Services Acquisition	72.1 - Does the application use third party components from trusted, Humana approved repositories?	Fully Implemented	Leveraging Kafka, Snap Logic and Databricks