User Story	Acceptance Criteria
As a Software company Customer, my data must be protected from unintentional disclosure to other customers or external parties.	Data is segregated by tenant. Administrators and users must be separated by role to prevent unauthorized disclosure or modification. Personally Identifiable Information (Software company RESTRICTED data) must be encrypted-at-rest and must be encrypted in transit over public networks.
As a Software company Customer, I need the application to allow passphrases and/or difficult passwords.	Verify password entry fields allow, or encourage, the use of passphrases, long passphrases or highly complex passwords. Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases.
As a Software company Customer, I need all connections to an application that contains my user data to be authenticated.	Verify that all connections to applications that contain customer information or functions are authenticated.
As a Software company Customer, I need password entry and other fields containing sensitive information to disallow caching or auto-complete.	Verify password and other data entry fields containing RESTRICTED information do not cache or allow auto-complete. An exception may be made for password managers.
As a Software company Customer, I need the ability to securely change or reset my password without worrying that my account(s) can be hijacked by a malicious party.	Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism. At a minimum, verify that the changing password functionality includes the old password, the new password, and a password confirmation. Verify that the forgotten password function and other recovery mechanisms *do not reveal the current password* and that the new password is not sent in clear text to the user. Verify that forgotten password and other recovery paths use a TOTP or other soft token, mobile push, or other offline recovery mechanism. Use of a random value in an e-mail or SMS should be a last resort.
As a Software company Customer, I want my account credentials to be created, stored and transported securely so that	Verify that account passwords are one way hashed with a salt, and there is sufficient work factor to defeat brute force and password hash recovery attacks. Verify that credentials are transported using a suitable encrypted

they can't be guessed, intercepted or reused by an attacker.	link and that all pages/functions that require a user to enter credentials are done so using an encrypted link.
As a Software company Customer, I need an additional factor of authentication to protect my accounts from unauthorized access. I want step-up authentication for risky transactions and changes to accounts.	Users can authenticate using two-factor authentication or other strong authentication, or any similar scheme that provides protection against username + password disclosure. Verify that risk based re-authentication, two factor or transaction signing is in place for high value transactions.
As a Software company Customer, I want a user logout feature. I need user sessions inactivated after logout and timed out after inactivity or a time limit set by an administrator.	Sessions are invalidated when the user logs out. Verify that sessions timeout after a specified period of inactivity or after an administratively- configurable maximum time period regardless of activity (an absolute timeout).
As a Software company Customer, application sessions must be unique and resistant to hijacking by a malicious actor.	All successful authentication and re- authentication generates a new session and session id. The session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies. Validate that only session ids generated by the application framework are recognized as active by the application. Confirm that session ids are sufficiently long, random and unique across the correct active session base. Session ids stored in cookies must have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the "HttpOnly" and "secure" attributes. Ensure that the application limits the number of active concurrent sessions and that an active session list is displayed in the account profile or similar of each user. The user should be able to terminate any active session. Verify the user is prompted with the option to terminate all other active sessions after a successful change password process.
As a Software company Customer, accounts must only be able to perform those actions or access resources explicitly granted to them.	The principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. Access to sensitive records is restricted, such that only authorized objects or data is accessible to each user (for example, protect

	against users tampering with a parameter to see or alter another user's account). If the application is multi-tenant, this segregation of users and access must be verified. The application must use strong random anti-CSRF tokens or has another transaction protection mechanism.
As a Software company Customer, I want the application to validate input to be correct and fit for the intended purpose.	Validate all data input from an external entity or client.
As a Software company Customer, I need access to application secret keys managed securely.	There is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired). Verify that this key lifecycle is properly enforced. Ensure that consumers of cryptographic services do not have direct access to key material. Isolate cryptographic processes, including master secrets and consider the use of a virtualized or physical hardware key vault (HSM).
As a Software company Customer, I need a suitable level of entropy in generating keys to prevent attacks.	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be not guessable by an attacker.
As a Software company Customer, I want the ability to collect logs in a standard format for a SIEM or other security tool, which contain information such as user and administrative access, but not sensitive information such as passwords or PII.	The application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id, software/framework versions and personal information. The application does not log sensitive data as defined under local privacy laws or regulations, organizational sensitive data as defined by a risk assessment, or sensitive authentication data that could assist an attacker, including user's session identifiers, passwords, hashes, or API tokens. Security logging controls provide the ability to log success and particularly failure events that are identified as security-relevant. Each log event should include necessary information that would allow for a detailed investigation of the timeline when an event happens.
As a Software company Customer, I want the ability to specify retention policies for certain RESTRICTED or sensitive data so	Verify that there is a method to remove each type of sensitive data from the application at the end of the required retention policy.

that this data can be deleted at the end of the retention period.	
As a Software company Customer, I want the application to detect and alert when there are unauthorized attempts to access my data, login or make changes.	The ability to limit the number of login attempts or send alerts when thresholds have been exceeded. Ability to send alerts when a role attempts an action beyond privilege level or if attempts to exceed privilege level hit a threshold.
As a Software company Customer, I want the application to provide an audit trail of administrative actions, user modifications or accessing PII.	Access or modifications to sensitive data is logged, if the data is collected under relevant data protection directives or where logging of accesses is required.
As a Software company Customer, I need web applications to be resistant to attacks by malicious actors that would impact availability or extract user data and/or credentials.	Verify that the application uses appropriate methods such as GET or POST, that safe character sets are used, same-origin policies are observed.
As a Software company Customer, if an application component is vulnerable, I want the malicious activity against it to have minimal or no impact on the rest of the application.	All malicious activity is adequately sandboxed, containerized or isolated to delay and deter attackers from attacking other applications.
As a Software company Customer, I want the application to be free of back doors, Easter eggs, and logic flaws, which could be abused by an attacker.	Verify that application source code and 3rd party libraries do not contain back doors, Easter eggs, and logic flaws in authentication, access control, input validation, and the business logic of high value transactions.
As a Software company Customer, I want the application to use business logic with a sequentially ordered flow that includes limits to prevent automated attacks against financial modules and PII.	Validate the application will only process business logic flows in sequential step order, with all steps being processed in realistic human time, and not process out of order, skipped steps, process steps from another user, or too quickly submitted transactions. Ensure the application has business limits and correctly enforces on a per user basis, with configurable alerting and automated reactions to automated or unusual attack.
As a Software company Customer, I want the application to handle untrusted data securely.	Untrusted file data should be handled in a secure manner, with data from untrusted sources stored outside the webroot and with limited permissions. Verify that files obtained from untrusted sources are validated to be

	of expected type. Untrusted file data submitted to the application is not used directly with file I/O commands, particularly to protect against path traversal, local file include, file mime type, and OS command injection vulnerabilities. Untrusted data is not used within inclusion, class loader, or reflection capabilities to prevent remote/local file inclusion vulnerabilities. Untrusted data is not used within cross- domain resource sharing (CORS) to protect against arbitrary remote content. The application code does not execute uploaded data obtained from untrusted sources.
As a Software company Customer, I want the application to enforce the Same-Origin Policy for content.	Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content. the web or application server is configured by default to deny access to remote resources or systems outside the web or application server. Implement a Content Security Policy, where possible.
As a Software company Customer, I want my mobile client to enforce the same level of security controls as other Software company applications.	Mobile clients must adhere to the Software company Secure Software Development Standard and are assessed with the same level of rigor as other Software company applications.
As a Software company Customer, I want the mobile client to protect user data.	The mobile app does not store sensitive data onto potentially unencrypted shared resources on the device (e.g. SD card or shared folders). Verify that sensitive data is not stored unprotected on the device, even in system protected areas such as key chains. The mobile app prevents leaking of sensitive information (for example, screenshots are saved of the current application as the application is backgrounded or writing sensitive information in console).
As a Software company Customer, I want the mobile client to use secure authentication methods.	ID values stored on the device and retrievable by other applications, such as the UDID or IMEI number are not used as authentication tokens. Verify that secret keys, API tokens, or passwords are dynamically generated in mobile applications.

As a Software company Customer, I want the mobile client to observe the principle of "least privilege."	The application requests minimal permissions for required functionality and resources.
As a Software company Customer, I want the mobile client to use secure coding techniques.	The application sensitive code is laid out unpredictably in memory (For example ASLR) and that there are anti-debugging techniques present that are sufficient to deter or delay likely attackers from injecting debuggers into the mobile app (For example GDB). The app does not export sensitive activities, intents, or content providers for other mobile apps on the same device to exploit. Sensitive information maintained in memory is overwritten with zeros as soon as it no longer required, to mitigate memory dumping attacks. Verify that the app validates input to exported activities, intents, or content providers.
As a Software company Customer, I want the web service to implement proper authentication, authorization and session management.	Access to administration and management functions within the Web Service Application is limited to web service administrators. Verify the use of session-based authentication and authorization. Avoid the use of static "API keys" and similar.
As a Software company Customer, I want the web service to validate input.	The same encoding style is used between the client and the server. Confirm that XML or JSON schema is in place and verified before accepting input. Ensure that all input is limited to an appropriate size limit. A REST service explicitly checks the incoming Content-Type to be the expected one, such as application/xml or application/json.
As a Software company Customer, I want the web service to ensure confidentiality and integrity of the communication.	Verify that SOAP based web services are compliant with Web Services-Interoperability (WS-I) Basic Profile at minimum. This essentially means TLS encryption. Message payloads are signed to ensure reliable transport between client and service, using JSON Web Signing or WS-Security for SOAP requests. Verify that alternative and less secure access paths do not exist.
As a Software company Customer, I want a REST service protected from Cross-Site Request Forgery (CSRF) attacks.	Use at one or more of the following: ORIGIN checks, double submit cookie pattern, CSRF nonces, and referrer checks.

As a Software company Customer, I want all components of the application to have up-to-date libraries and platforms.	All components should be up to date with proper security configuration(s) and version(s). Verify that third party components come from trusted repositories.
As a Software company Customer, I want the application configuration to be “secure by default.”	Remove unneeded configurations and folders such as sample applications, platform documentation, and default or example users. Communications between components, such as between the application server and the database server, should be encrypted, particularly when the components are in different containers or on different systems. Communications between components, such as between the application server and the database server should be authenticated using an account with the least necessary privileges. Verify that all application components are signed and that build processes for system level languages have all security flags enabled, such as ASLR, DEP, and security checks.
As a Software company Customer, I want the application secured and hardened by default so that user changes do not expose security vulnerabilities or flaws with underlying systems.	Validate that application deployments are adequately sandboxed, containerized or isolated to delay and deter attackers from attacking other applications. Verify that the application build and deployment processes are performed in a secure fashion. Confirm all application assets are hosted by the application rather than on a CDN or external provider, including; JavaScript libraries, CSS stylesheets and web fonts.

Security Acceptance Criteria

More often, existing user stories will contain security requirements as acceptance criteria. Examples follow below.

Story or Feature Topic	Security Acceptance Criteria
Application Architecture and Design	Data flow and process flow diagrams are current and assessed for risk by Information Security
Application Architecture and Design	System and application components are hardened according to Information Security requirements and checked for vulnerabilities through manual or automated security testing. All critical and high vulnerabilities will be remediated prior to release.
Application Architecture and Design	Components are segregated based on compliance requirements and Information Security Data Handling Requirements.
Authentication, Authorization, Accounting	All pages and resources require authentication except those specifically intended to be public.
Authentication, Authorization, Accounting	Forms containing credentials are not filled in by the application. Pre-populating a form could mean that credentials are stored in plaintext or a reversible format.
Authentication, Authorization, Accounting	All authentication controls are enforced on the server side.
Authentication, Authorization, Accounting	All authentication controls fail securely to ensure attackers cannot log in.
Authentication, Authorization, Accounting	All authentication decisions can be logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.
Authentication, Authorization, Accounting	Information enumeration is not possible via login, password reset, or forgot account functionality.
Authentication, Authorization, Accounting	Verify there are no default passwords in use for the application framework or any components used by the application (such as “admin/password”).
Authentication, Authorization, Accounting	Anti-automation is in place to prevent breached credential testing, brute forcing, and account lockout attacks.
Authentication, Authorization, Accounting	All authentication credentials for accessing services external to the application are encrypted and stored in a protected location.

Authentication, Authorization, Accounting	Account lockout is divided into soft and hard lock status, and these are not mutually exclusive. If an account is temporarily soft locked out due to a brute force attack, this should not reset the hard lock status.
Authentication, Authorization, Accounting	If shared knowledge based questions (also known as "secret questions") are required, the questions must not violate privacy laws and are sufficiently strong to protect accounts from malicious recovery.
Authentication, Authorization, Accounting	The system can be configured to disallow the use of a configurable number of previous passwords.
Authentication, Authorization, Accounting	All authentication challenges, whether successful or failed, should respond in the same average response time.
Authentication, Authorization, Accounting	Secrets, API keys, and passwords must not be included in the source code, or online source code repositories.
Authentication, Authorization, Accounting	Administrative interfaces are not accessible to untrusted parties.
Access Control	Directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.
Access Control	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.
Access Control	A centralized mechanism (including libraries that call external authorization services) is used for protecting access to each type of protected resource.
Access Control	All access control decisions can be logged and all failed decisions are logged.
Access Control	The application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.

Access Control	The application correctly enforces context- sensitive authorization to not allow unauthorized manipulation by means of parameter tampering.
Input Handling Verification	The runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.
Input Handling Verification	Input validation routines are enforced on the server side and input validation failures result in request rejection and are logged.
Input Handling Verification	A single input validation control is used by the application for each type of data that is accepted.
Input Handling Verification	All SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected using prepared statements or query parameterization, and thus not susceptible to SQL injection
Input Handling Verification	The application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection.
Input Handling Verification	The application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.
Input Handling Verification	The application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.
Input Handling Verification	The application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.
Input Handling Verification	Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilize templates that automatically encode contextually to ensure the application is not susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks.
Input Handling Verification	If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the inbound request to a model, verify that security sensitive fields such as “*accountBalance*”, “*role*” or “*password*” are protected from malicious automatic binding.
Input Handling Verification	Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no

	distinction about the source of request parameters (GET, POST, cookies, headers, environment, etc.)
Input Handling Verification	All input data is validated, not only HTML form fields but all sources of input such as REST calls, query parameters, HTTP headers, cookies, batch files, RSS feeds, etc; using positive validation (whitelisting), then lesser forms of validation such as greylisting (eliminating known bad strings), or rejecting bad inputs (blacklisting).
Input Handling Verification	Structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as validating suburbs and zip or post codes match).
Input Handling Verification	Unstructured data is sanitized to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped (e.g. natural names with Unicode or apostrophes, such as ねこ or O'Hara)
Input Handling Verification	Untrusted HTML from WYSIWYG editors or similar are properly sanitized with an HTML sanitizer and handle it appropriately according to the input validation task and encoding task.
Input Handling Verification	For auto-escaping template technology, if UI escaping is disabled, ensure that HTML sanitization is enabled instead.
Input Handling Verification	Data transferred from one DOM context to another, uses safe JavaScript methods, such as using .innerText and .val.
Input Handling Verification	When parsing JSON in browsers, that JSON.parse is used to parse JSON on the client. Do not use eval() to parse JSON on the client.
Input Handling Verification	Authenticated data is cleared from client storage, such as the browser DOM, after the session is terminated.
Encryption-at-rest	All cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding.
Encryption-at-rest	Cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard.
Encryption-at-rest	Sensitive passwords or key material maintained in memory is overwritten with zeros as soon as it no longer required, to mitigate memory dumping attacks.

Encryption-at-rest	Verify that all keys and passwords are replaceable, and are generated or replaced at installation time.
Encryption-at-rest	Random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.
Error Handling and Logging	Security logs are protected from unauthorized access and modification.
Error Handling and Logging	All non-printable symbols and field separators are properly encoded in log entries, to prevent log injection.
Error Handling and Logging	Log fields from trusted and untrusted sources are distinguishable in log entries and audit log or similar allows for non-repudiation of key transactions.
Error Handling and Logging	The logs are stored on a different partition than the application is running with proper log rotation.
Error Handling and Logging	Time sources should be synchronized to ensure logs have the correct time.
Data Protection and Verification	List classification of data processed by the application according to Software Company's Information Classification Standard. This should be documented in data flow diagrams. There must be explicit enforcement in accordance with the Software company Data Handling Matrix requirements.
Data Protection and Verification	Verify all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data).
Data Protection and Verification	The application sets appropriate anti-caching headers as per the risk tier of the application and data it handles.
Data Protection and Verification	On the server, all cached or temporary copies of sensitive data stored are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.
Data Protection and Verification	Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.

Data Protection and Verification	Data stored in client side storage (such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies) does not contain sensitive data or PII.
Data Protection and Verification	Sensitive information maintained in memory is overwritten with zeros as soon as it no longer required, to mitigate memory dumping attacks.
Communications Security	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid. Also ensure that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information.
Communications Security	Appsec-approved version of TLS and set of cipher suites is used for connections (including both external and backend connections) that pass data classified as RESTRICTED or CONFIDENTIAL according to the Software company Information Classification Standard on public networks, and that it does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm.
Communications Security	Ensure that a single standard TLS implementation is used by the application and components, that it is configured in accordance with Information Security’s Encryption Standard.
HTTP security configuration verification	Application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods (e.g. TRACE, PUT, and DELETE) are explicitly blocked.
HTTP security configuration verification	Every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).
HTTP security configuration verification	HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.
HTTP security configuration verification	Suitable X-FRAME-OPTIONS headers are in use for sites where content should not be viewed in a 3rd-party X-Frame.
HTTP security configuration verification	HTTP headers or any part of the HTTP response do not expose detailed version information of system components.
HTTP security configuration verification	Verify that all API responses contain X-Content- Type-Options: nosniff and Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).

HTTP security configuration verification	Content security policy (CSPv2) is in place that helps mitigate common DOM, XSS, JSON, and JavaScript injection vulnerabilities.
HTTP security configuration verification	X-XSS-Protection: 1; mode=block header is in place to enable browser reflected XSS filters.
Files and resources verification	Avoid Flash, Active-X, Silverlight, NACL, client- side Java or other client side technologies not supported natively via W3C browser standards.
Files and resources verification	Enforce Same-Origin Policy and/or implement Content Security Policy to protect against XSS, clickjacking and other code injection attacks.