



Jeff Murray's Programming Shop, Inc.

Template for CAREWare Stakeholder Risk Assessment Aligned with Application Security Verification Standard (ASVS) 2025

9/25/2025

Overview	18
V1 Architecture, Design and Threat Modeling	18
V1.1 Secure Software Development Lifecycle	19
1.1.1 Verify the use of a secure software development lifecycle that addresses security in all stages of development.	19
1.1.2 Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.	20
Threat Boundaries Diagram and Notes	21
Threat Identification for HTTP Server Boundary	21
Threat Identification for HTTP Server OS File Boundary	22
Threat Identification for Business Tier Boundary	22
Threat Identification for Business Tier OS File Boundary	23
Threat Identification for SQL Server Boundary	23
Data Flow Diagram for HTTP Server Boundary and Notes	24
CAREWare's API URL prefixes	25
Data Flow Diagram for HTTP Server OS File Boundary	27
Data Flow Diagram for Business Tier Boundary	30
Data Flow Diagram for Business Tier OS File Boundary	32
Data Flow Diagram for SQL Server Boundary	38
1.1.3 Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or	

edit anyone else's profile"	39
1.1.4 Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.	40
1.1.6 Verify implementation of centralized, simple (economy of design), vetted, secure, and reusable security controls to avoid duplicate, missing, ineffective, or insecure controls.	40
1.1.7 Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers.	40
V1.2 Authentication Architecture	40
1.2.1 Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.	40
1.2.2 Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed.	41
1.2.3 Verify that the application uses a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches.	42
1.2.4 Verify that all authentication pathways and identity management APIs implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application	42
V1.3 Session Management Architecture. This is a placeholder for future architectural requirements.	43
V1.4 Access Control Architecture	43
1.4.1 Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions enforce access controls. Never enforce access controls on the client.	43
1.4.2 [DELETED, NOT ACTIONABLE]	43
1.4.3 [DELETED, DUPLICATE OF 4.1.2]	43
1.4.4 Verify the application uses a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths.	43
1.4.5 Verify that attribute or feature-based access control is used whereby the code checks the user's authorization for a feature/data item rather than just their role. Permissions should still be allocated using roles.	43
V1.5 Input and Output Architecture	44
1.5.1 Verify that input and output requirements clearly define how to handle and process data based on type, content, and applicable laws, regulations, and other policy compliance.	44
1.5.2 Verify that serialization is not used when communicating with untrusted clients. If this is not possible, ensure that adequate integrity controls (and possibly encryption if sensitive data is sent) are enforced to prevent deserialization attacks including object injection.	45
1.5.3 Verify that input validation is enforced on a trusted service layer.	45
1.5.4 Verify that output encoding occurs close to or by the interpreter for which it is intended.	45
V1.6 Cryptographic Architecture	45

1.6.1 Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57	45
1.6.2 Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives.	46
1.6.3 Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.	46
1.6.4 Verify that the architecture treats client-side secrets--such as symmetric keys, passwords, or API tokens--as insecure and never uses them to protect or access sensitive data.	46
V1.7 Errors, Logging and Auditing Architecture	47
1.7.1 Verify that a common logging format and approach is used across the system.	47
1.7.2 Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation.	47
V1.8 Data Protection and Privacy Architecture	47
1.8.1 Verify that all sensitive data is identified and classified into protection levels.	48
1.8.2 Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture.	48
V1.9 Communications Architecture	49
1.9.1 Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers.	49
1.9.2 Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For example, application components should validate TLS certificates and chains.	49
V1.10 Malicious Software Architecture	50
1.10.1 Verify that a source code control system is in use, with procedures to ensure that check-ins are accompanied by issues or change tickets. The source code control system should have access control and identifiable users to allow traceability of any changes.	50
V1.11 Business Logic Architecture	50
1.11.1 Verify the definition and documentation of all application components in terms of the business or security functions they provide.	50
1.11.2 Verify that all high-value business logic flows, including authentication, session management and access control, do not share unsynchronized state.	50
1.11.3 Verify that all high-value business logic flows, including authentication, session management and access control are thread safe and resistant to time-of-check and time-of-use race conditions.	51
V1.12 Secure File Upload Architecture	51
1.12.1 [DELETED, DUPLICATE OF 12.4.1]	51
1.12.2 Verify that user-uploaded files - if required to be displayed or downloaded from the application - are served by either octet stream downloads, or from an unrelated domain, such as a cloud file storage bucket. Implement a suitable Content Security Policy (CSP) to reduce the risk from XSS vectors or other attacks from the uploaded file.	51
V1.13 API Architecture	51

V1.14 Configuration Architecture	51
1.14.1 Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms.	51
1.14.2 Verify that binary signatures, trusted connections, and verified endpoints are used to deploy binaries to remote devices.	52
1.14.3 Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.	52
1.14.4 Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts.	53
1.14.5 Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization.	53
1.14.6 Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets.	53
V2 Authentication	54
V2.1 Password Security	54
2.1.1 Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined). (C6)	54
2.1.2 Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied. (C6)	54
2.1.3 Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space. (C6)	55
2.1.4 Verify that any printable Unicode character, including language-neutral characters such as spaces and emojis, are permitted in passwords.	55
2.1.6 Verify that password change functionality requires the user's current and new password.	55
2.1.7 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. (C6)	56
2.1.8 Verify that a password strength meter is provided to help users set a stronger password.	56
2.1.9 Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters. (C6)	56
2.1.10 Verify that there are no periodic credential rotation or password history requirements	57
2.1.11 Verify that "paste" functionality, browser password helpers, and external password managers are permitted	57

2.1.12 Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as native functionality	57
V2.2 General Authenticator Requirements	57
2.2.1 Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account.	57
2.2.2 Verify that the use of weak authenticators (such as SMS and email) is limited to secondary verification and transaction approval and not as a replacement for more secure authentication methods. Verify that stronger methods are offered before weak methods, users are aware of the risks, or that proper measures are in place to limit the risks of account compromise.	58
2.2.3 Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification.	58
2.2.4 Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates	58
2.2.5 Verify that where a credential service provider (CSP) and the application verifying authentication are separated, mutually authenticated TLS is in place between the two endpoints.	59
2.2.6 Verify replay resistance through the mandated use of OTP devices, cryptographic authenticators, or lookup codes.	59
2.2.7 Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such as a button press on a FIDO hardware key.	59
V2.3 Authenticator Lifecycle Requirements	59
2.3.1 Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long term password.	59
2.3.2 Verify that enrollment and use of subscriber-provided authentication devices are supported, such as a U2F or FIDO tokens.	59
2.3.3 Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.	60
V2.4 Credential Storage Requirements	60
2.4.1 Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved oneway key derivation or password hashing function. Key derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash. (C6)	60
2.4.2 Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored. (C6)	60

2.4.3 Verify that if PBKDF2 is used, the iteration count SHOULD be as large as verification server performance will allow, typically at least 100,000 iterations. (C6)	60
2.4.4 Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, typically at least 13. (C6)	61
2.4.5 Verify that an additional iteration of a key derivation function is performed, using a salt value that is secret and known only to the verifier. Generate the salt value using an approved random bit generator [SP 800-90Ar1] and provide at least the minimum security strength specified in the latest revision of SP 800-131A. The secret salt value SHALL be stored separately from the hashed passwords (e.g., in a specialized device like a hardware security module).	61
V2.5 Credential Recovery Requirements	61
2.5.1 Verify that a system generated initial activation or recovery secret is not sent in clear text to the user. (C6)	61
2.5.2 Verify password hints or knowledge-based authentication (so-called "secret questions") are not present.	61
2.5.3 Verify password credential recovery does not reveal the current password in any way. (C6)	62
2.5.4 Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").	62
2.5.5 Verify that if an authentication factor is changed or replaced, that the user is notified of this event.	62
2.5.6 Verify forgotten password, and other recovery paths use a secure recovery mechanism, such as TOTP or other soft token, mobile push, or another offline recovery mechanism. (C6)	63
2.5.7 Verify that if OTP or multi-factor authentication factors are lost, that evidence of identity proofing is performed at the same level as during enrollment.	63
V2.6 Look-up Secret Verifier Requirements	63
2.6.1 Verify that lookup secrets can be used only once.	63
2.6.2 Verify that lookup secrets have sufficient randomness (112 bits of entropy), or if less than 112 bits of entropy, salted with a unique and random 32-bit salt and hashed with an approved one-way hash.	63
2.6.3 Verify that lookup secrets are resistant to offline attacks, such as predictable values.	64
V2.7 Out of Band Verifier Requirements	64
2.7.1 Verify that clear text out of band (NIST "restricted") authenticators, such as SMS or PSTN, are not offered by default, and stronger alternatives such as push notifications are offered first.	64
2.7.2 Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.	64
2.7.3 Verify that the out of band verifier authentication requests, codes, or tokens are only usable once, and only for the original authentication request.	64
2.7.4 Verify that the out of band authenticator and verifier communicates over a secure independent channel.	65
2.7.5 Verify that the out of band verifier retains only a hashed version of the authentication code.	65
2.7.6 Verify that the initial authentication code is generated by a secure random number generator, containing at least 20 bits of entropy (typically a six digit random number is	

sufficient).	65
V2.8 Single or Multi Factor One Time Verifier Requirements	65
2.8.1 Verify that time-based OTPs have a defined lifetime before expiring.	65
2.8.2 Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage.	65
2.8.3 Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.	66
2.8.4 Verify that time-based OTP can be used only once within the validity period.	66
2.8.5 Verify that if a time-based multi factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device.	66
2.8.6 Verify physical single factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location.	66
2.8.7 Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know.	66
V2.9 Cryptographic Software and Devices Verifier Requirements	66
2.9.1 Verify that cryptographic keys used in verification are stored securely and protected against disclosure, such as using a TPM or HSM, or an OS service that can use this secure storage.	67
2.9.2 Verify that the challenge nonce is at least 64 bits in length, and statistically unique or unique over the lifetime of the cryptographic device.	67
2.9.3 Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.	67
V2.10 Service Authentication Requirements	67
2.10.1 Verify that integration secrets do not rely on unchanging passwords, such as API keys or shared privileged accounts.	67
2.10.2 Verify that if passwords are required, the credentials are not a default account.	67
2.10.3 Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access.	68
2.10.4 Verify passwords, integrations with databases and third-party systems, seeds and internal secrets, and API keys are managed securely and not included in the source code or stored within source code repositories. Such storage SHOULD resist offline attacks. The use of a secure software key store (L1), hardware trusted platform module (TPM), or a hardware security module (L3) is recommended for password storage.	68
V3: Session Management Verification Requirements	68
V3.1 Fundamental Session Management Requirements	68
3.1.1 Verify the application never reveals session tokens in URL parameters or error messages	68
V3.2 Session Binding Requirements	68
3.2.1 Verify the application generates a new session token on user authentication. (C6)	69
3.2.2 Verify that session tokens possess at least 64 bits of entropy. (C6)	69
3.2.3 Verify the application only stores session tokens in the browser using secure	

methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.	69
3.2.4 Verify that session token are generated using approved cryptographic algorithms. (C6)	69
V3.3 Session Logout and Timeout Requirements	69
3.3.1 Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties. (C6)	69
3.3.2 If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period. (C6)	69
3.3.3 Verify that the application terminates all other active sessions after a successful password change, and that this is effective across the application, federated login (if present), and any relying parties.	70
3.3.4 Verify that users are able to view and log out of any or all currently active sessions and devices.	70
V3.4 Cookie-based Session Management	70
3.4.1 Verify that cookie-based session tokens have the 'Secure' attribute set. (C6)	70
3.4.2 Verify that cookie-based session tokens have the 'HttpOnly' attribute set. (C6)	70
3.4.3 Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks. (C6)	70
3.4.4 Verify that cookie-based session tokens use "__Host-" prefix (see references) to provide session cookie confidentiality.	70
3.4.5 Verify that if the application is published under a domain name with other applications that set or use session cookies that might override or disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible. (C6)	71
V3.5 Token-based Session Management	71
3.5.1 Verify the application does not treat OAuth and refresh tokens — on their own — as the presence of the subscriber and allows users to terminate trust relationships with linked applications.	71
3.5.2 Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.	71
3.5.3 Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.	71
V3.6 Re-authentication from a Federation or Assertion	71
3.6.1 Verify that relying parties specify the maximum authentication time to CSPs and that CSPs re-authenticate the subscriber if they haven't used a session within that period.	72
3.6.2 Verify that CSPs inform relying parties of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.	72
V3.7 Defenses Against Session Management Exploits	72
3.7.1 Verify the application ensures a valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.	72
V4: Access Control Verification Requirements	72

4.1 General Access Control Design	72
4.1.1 Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.	72
4.1.2 Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	73
4.1.3 Verify that 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. (C7)	73
4.1.4 Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new features until access is explicitly assigned. (C7)	73
4.1.5 Verify that access controls fail securely including when an exception occurs. (C10)	74
V4.2 Operation Level Access Control	74
4.2.1 Verify that sensitive data and APIs are protected against direct object attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.	74
4.2.2 Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.	74
V4.3 Other Access Control Considerations	74
4.3.1 Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.	74
4.3.2 Verify that 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.	75
4.3.3 Verify 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.	75
V5: Validation, Sanitization and Encoding Verification Requirements	75
V5.1 Input Validation	75
5.1.1 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, or environment variables).	75
5.1.2 Verify that frameworks protect against mass parameter assignment attacks, or that the application has countermeasures to protect against unsafe parameter assignment, such as marking fields private or similar. (C5)	76
5.1.3 Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (whitelisting). (C5)	76
5.1.4 Verify that 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 checking that suburb and zip/postcode match). (C5)	76
5.1.5 Verify that URL redirects and forwards only allow whitelisted destinations, or show	

a warning when redirecting to potentially untrusted content.	76
V5.2 Sanitization and Sandboxing Requirements	76
5.2.1 Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature. (C5)	76
5.2.2 Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.	77
5.2.3 Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.	77
5.2.4 Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.	77
5.2.5 Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.	77
5.2.6 Verify that the application protects against SSRF attacks, by validating or sanitizing untrusted data or HTTP file metadata, such as filenames and URL input fields, use whitelisting of protocols, domains, paths and ports.	77
5.2.7 Verify that the application sanitizes, disables, or sandboxes user-supplied SVG scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.	77
5.2.8 Verify that the application sanitizes, disables, or sandboxes user-supplied scriptable or expression template language content, such as Markdown, CSS or XSL stylesheets, BBCode, or similar.	78
V5.3 Output encoding and Injection Prevention Requirements	78
5.3.1 Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL Parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara). (C4)	78
5.3.2 Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled. (C4)	78
5.3.3 Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS. (C4)	78
5.3.4 Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks. (C3)	78
5.3.5 Verify that where parameterized or safer mechanisms are not present, context-specific output encoding is used to protect against injection attacks, such as the use of SQL escaping to protect against SQL injection. (C3, C4)	79
5.3.7 Verify that the application protects against LDAP Injection vulnerabilities, or that specific security controls to prevent LDAP Injection have been implemented. (C4)	79
5.3.8 Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding. (C4)	79
5.3.9 Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.	79
5.3.10 Verify that the application protects against XPath injection or XML injection	

attacks. (C4)	79
V5.4 Memory, String, and Unmanaged Code Requirements	79
5.4.1 Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.	80
5.4.2 Verify that format strings do not take potentially hostile input, and are constant.	80
5.4.3 Verify that sign, range, and input validation techniques are used to prevent integer overflows.	80
V5.5 Deserialization Prevention Requirements	80
5.5.1 Verify that serialized objects use integrity checks or are encrypted to prevent hostile object creation or data tampering. (C5)	80
5.5.2 Verify that the application correctly restricts XML parsers to only use the most restrictive configuration possible and to ensure that unsafe features such as resolving external entities are disabled to prevent XXE.	80
5.5.3 Verify that deserialization of untrusted data is avoided or is protected in both custom code and third-party libraries (such as JSON, XML and YAML parsers).	80
5.5.4 Verify that when parsing JSON in browsers or JavaScript-based backends, JSON.parse is used to parse the JSON document. Do not use eval() to parse JSON.	81
V6: Stored Cryptography Verification Requirements	81
V6.1 Data Classification	81
6.1.1 Verify that regulated private data is stored encrypted while at rest, such as personally identifiable information (PII), sensitive personal information, or data assessed likely to be subject to EU's GDPR.	81
6.1.2 Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.	81
6.1.3 Verify that regulated financial data is stored encrypted while at rest, such as financial accounts, defaults or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records.	82
V6.2 Algorithms	82
6.2.1 Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.	83
6.2.2 Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography. (C8)	83
6.2.3 Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.	83
6.2.4 Verify that random number, encryption or hashing algorithms, key lengths, rounds, ciphers or modes, can be reconfigured, upgraded, or swapped at any time, to protect against cryptographic breaks. (C8)	83
6.2.6 Verify that nonces, initialization vectors, and other single use numbers must not be used more than once with a given encryption key. The method of generation must be appropriate for the algorithm being used.	83
6.2.7 Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.	84
V6.3 Random Values	84
6.3.1 Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved cryptographically	

secure random number generator when these random values are intended to be not guessable by an attacker.	84
6.3.2 Verify that random GUIDs are created using the GUID v4 algorithm, and a cryptographically-secure pseudo-random number generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable.	84
6.3.3 Verify that random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.	84
V6.4 Secret Management	85
6.4.1 Verify that a secrets management solution such as a key vault is used to securely create, store, control access to and destroy secrets. (C8)	85
6.4.2 Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations. (C8)	85
V7: Error Handling and Logging Verification Requirements	85
V7.1 Log Content Requirements	85
7.1.1 Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form. (C9, C10)	85
7.1.2 Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy. (C9)	85
7.1.3 Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures. (C5, C7)	86
7.1.4 Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens. (C9)	86
V7.2 Log Processing Requirements	86
7.2.1 Verify that all authentication decisions are logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.	86
7.2.2 Verify that all access control decisions can be logged and all failed decisions are logged. This should include requests with relevant metadata needed for security investigations.	86
V7.3 Log Protection Requirements	86
7.3.1 Verify that the application appropriately encodes user-supplied data to prevent log injection. (C9)	86
7.3.2 Verify that all events are protected from injection when viewed in log viewing software. (C9)	87
7.3.3 Verify that security logs are protected from unauthorized access and modification. (C9)	87
7.3.4 Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis. (C9)	87
V7.4 Error Handling	88
7.4.1 Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate. (C10)	88
7.4.2 Verify that exception handling (or a functional equivalent) is used across the	

codebase to account for expected and unexpected error conditions. (C10)	88
7.4.3 Verify that a "last resort" error handler is defined which will catch all unhandled exceptions. (C10)	88
V8: Data Protection Verification Requirements	88
8.1 General Data Protection	89
8.1.1 Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.	89
8.1.2 Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.	89
8.1.3 Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.	89
8.1.4 Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.	89
8.1.5 Verify that regular backups of important data are performed and that test restoration of data is performed.	90
8.1.6 Verify that backups are stored securely to prevent data from being stolen or corrupted.	90
V8.2 Client-side Data Protection	90
8.2.1 Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.	90
8.2.2 Verify that 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.	91
8.2.3 Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.	91
V8.3 Sensitive Private Data	91
8.3.1 Verify that sensitive data is sent to the server in the HTTP message body or headers, and that query string parameters from any HTTP verb do not contain sensitive data.	91
8.3.2 Verify that users have a method to remove or export their data on demand.	91
8.3.3 Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.	91
8.3.4 Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data. (C8)	92
8.3.5 Verify accessing sensitive data is audited (without logging the sensitive data itself), if the data is collected under relevant data protection directives or where logging of access is required.	92
8.3.6 Verify that sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.	93
8.3.7 Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity. (C8)	93
8.3.8 Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the	

situation requires.	93
V9: Communications Verification Requirements	94
V9.1 Communications Security Requirements	94
9.1.1 Verify that secured TLS is used for all client connectivity, and does not fall back to insecure or unencrypted protocols. (C8)	94
9.1.2 Verify using online or up to date TLS testing tools that only strong algorithms, ciphers, and protocols are enabled, with the strongest algorithms and ciphers set as preferred.	94
9.1.3 Verify that old versions of SSL and TLS protocols, algorithms, ciphers, and configuration are disabled, such as SSLv2, SSLv3, or TLS 1.0 and TLS 1.1. The latest version of TLS should be the preferred cipher suite.	94
V9.2 Server Communications Security Requirements	95
9.2.1 Verify that connections to and from the server use trusted TLS certificates. Where internally generated or self-signed certificates are used, the server must be configured to only trust specific internal CAs and specific self-signed certificates. All others should be rejected.	95
9.2.2 Verify that encrypted communications such as TLS is used for all inbound and outbound connections, including for management ports, monitoring, authentication, API, or web service calls, database, cloud, serverless, mainframe, external, and partner connections. The server must not fall back to insecure or unencrypted protocols.	95
9.2.3 Verify that all encrypted connections to external systems that involve sensitive information or functions are authenticated.	96
9.2.4 Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.	96
9.2.5 Verify that backend TLS connection failures are logged.	96
V10: Malicious Code Verification Requirements	96
V10.1 Code Integrity Controls	96
10.1.1 Verify that a code analysis tool is in use that can detect potentially malicious code, such as time functions, unsafe file operations and network connections.	96
V10.2 Malicious Code Search	97
10.2.1 Verify that the application source code and third party libraries do not contain unauthorized phone home or data collection capabilities. Where such functionality exists, obtain the user's permission for it to operate before collecting any data.	97
10.2.2 Verify that the application does not ask for unnecessary or excessive permissions to privacy related features or sensors, such as contacts, cameras, microphones, or location.	97
10.2.3 Verify that the application source code and third party libraries do not contain back doors, such as hard-coded or additional undocumented accounts or keys, code obfuscation, undocumented binary blobs, rootkits, or anti-debugging, insecure debugging features, or otherwise out of date, insecure, or hidden functionality that could be used maliciously if discovered.	97
10.2.4 Verify that the application source code and third party libraries does not contain time bombs by searching for date and time related functions.	97
10.2.5 Verify that the application source code and third party libraries does not contain malicious code, such as salami attacks, logic bypasses, or logic bombs.	97
No stakeholder threats identified	98

10.2.6 Verify that the application source code and third party libraries do not contain Easter eggs or any other potentially unwanted functionality.	98
V10.3 Deployed Application Integrity Controls	98
10.3.1 Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.	98
10.3.2 Verify that the application employs integrity protections, such as code signing or sub-resource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.	98
10.3.3 Verify that the application has protection from sub-domain takeovers if the application relies upon DNS entries or DNS sub-domains, such as expired domain names, out of date DNS pointers or CNAMEs, expired projects at public source code repos, or transient cloud APIs, serverless functions, or storage buckets (autogen-bucket-id.cloud.example.com) or similar. Protections can include ensuring that DNS names used by applications are regularly checked for expiry or change.	99
V11: Business Logic Verification Requirements	99
V11.1 Business Logic Security Requirements	99
11.1.1 Verify the application will only process business logic flows for the same user in sequential step order and without skipping steps.	99
11.1.2 Verify the application will only process business logic flows with all steps being processed in realistic human time, i.e. transactions are not submitted too quickly.	99
11.1.3 Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.	100
11.1.4 Verify the application has sufficient anti-automation controls to detect and protect against data exfiltration, excessive business logic requests, excessive file uploads or denial of service attacks.	100
11.1.5 Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modelling or similar methodologies.	100
11.1.6 Verify the application does not suffer from "time of check to time of use" (TOCTOU) issues or other race conditions for sensitive operations.	100
11.1.7 Verify the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt. (C9)	100
V12: File and Resources Verification Requirements	101
V12.1 File Upload Requirements	101
12.1.1 Verify that the application will not accept large files that could fill up storage or cause a denial of service attack.	101
12.1.2 Verify that compressed files are checked for "zip bombs" - small input files that will decompress into huge files thus exhausting file storage limits.	101
12.1.3 Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.	101
V12.2 File Integrity Requirements	102
12.2.1 Verify that files obtained from untrusted sources are validated to be of expected	

type based on the file's content.	102
V12.3 File execution Requirement	102
12.3.4 Verify that the application protects against reflective file download (RFD) by validating or ignoring user-submitted filenames in a JSON, JSONP, or URL parameter, the response Content-Type header should be set to text/plain, and the Content-Disposition header should have a fixed filename.	102
12.3.5 Verify that untrusted file metadata is not used directly with system API or libraries, to protect against OS command injection.	102
12.3.6 Verify that the application does not include and execute functionality from untrusted sources, such as unverified content distribution networks, JavaScript libraries, node npm libraries, or server-side DLLs.	103
V12.4 File Storage Requirements	103
12.4.1 Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions, preferably with strong validation.	103
12.4.2 Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload of known malicious content.	103
V12.5 File Download Requirements	104
12.5.1 Verify that the web tier is configured to serve only files with specific file extensions to prevent unintentional information and source code leakage. For example, backup files (e.g. .bak), temporary working files (e.g. .swp), compressed files (.zip, .tar.gz, etc) and other extensions commonly used by editors should be blocked unless required.	104
12.5.2 Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.	104
V12.6 SSRF Protection Requirements	104
12.6.1 Verify that the web or application server is configured with a whitelist of resources or systems to which the server can send requests or load data/files from.	104
V13: API and Web Service Verification Requirements	104
V13.1 Generic Web Service Security Verification Requirements	104
13.1.1 Verify that all application components use the same encodings and parsers to avoid parsing attacks that exploit different URI or file parsing behavior that could be used in SSRF and RFI attacks.	105
No stakeholder threats identified	105
13.1.2 Verify that access to administration and management functions is limited to authorized administrators.	105
13.1.3 Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.	105
No stakeholder threats identified	105
13.1.4 Verify that authorization decisions are made at both the URI, enforced by programmatic or declarative security at the controller or router, and at the resource level, enforced by model-based permissions.	105
13.1.5 Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).	106
V13.2 RESTful Web Service Verification Requirements	106
CAREWare is not a RESTful Web Service	106

V13.3 SOAP Web Service Verification Requirements	106
CAREWare is not a SOAP Web Service	106
V13.4 GraphQL and other Web Service Data Layer Security Requirements	107
CAREWare is not a GraphQL Web Service	107
V14: Configuration Verification Requirements	107
V14.1 Build	107
14.1.1 Verify that the application build and deployment processes are performed in a secure and repeatable way, such as CI / CD automation, automated configuration management, and automated deployment scripts.	107
14.1.2 Verify that compiler flags are configured to enable all available buffer overflow protections and warnings, including stack randomization, data execution prevention, and to break the build of an unsafe pointer, memory, format string, integer, or string operations are found.	107
14.1.3 Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.	107
14.1.4 Verify that the application, configuration, and all dependencies can be redeployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.	108
14.1.5 Verify that authorized administrators can verify the integrity of all security relevant configurations to detect tampering.	108
V14.2 Dependency	109
14.2.1 Verify that all components are up to date, preferably using a dependency checker during build or compile time. (C2)	109
14.2.2 Verify that all unneeded features, documentation, samples, configurations are removed, such as sample applications, platform documentation, and default or example users.	109
14.2.3 Verify that if application assets, such as JavaScript libraries, CSS stylesheets or web fonts, are hosted externally on a content delivery network (CDN) or external provider, Subresource Integrity (SRI) is used to validate the integrity of the asset.	109
14.2.4 Verify that third party components come from pre-defined, trusted and continually maintained repositories. (C2)	109
14.2.5 Verify that an inventory catalog is maintained of all third party libraries in use. (C2)	110
14.2.6 Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behavior into the application. (C2)	110
V14.3 Unintended Security Disclosure Requirements	110
14.3.1 Verify that web or application server and framework error messages are configured to deliver user actionable, customized responses to eliminate any unintended security disclosures.	110
14.3.2 Verify that web or application server and application framework debug modes are disabled in production to eliminate debug features, developer consoles, and unintended security disclosures.	110
14.3.3 Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.	110
V14.4 HTTP Security Headers Requirements	110

14.4.1 Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1)	111
14.4.2 Verify that all API responses contain Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).	111
14.4.3 Verify that a content security policy (CSPv2) is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.	111
14.4.4 Verify that all responses contain X-Content-Type-Options: nosniff.	111
14.4.5 Verify that HTTP Strict Transport Security headers are included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.	111
14.4.6 Verify that a suitable "Referrer-Policy" header is included, such as "no-referrer" or "same-origin"	111
14.4.7 Verify that a suitable X-Frame-Options or Content-Security-Policy: frame-ancestors header is in use for sites where content should not be embedded in a third-party site.	111
V14.5 Validate HTTP Request Header Requirements	112
14.5.1 Verify that the application server only accepts the HTTP methods in use by the application or API, including pre-flight OPTIONS.	112
14.5.2 Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.	112
14.5.3 Verify that the cross-domain resource sharing (CORS) Access-Control-Allow-Origin header uses a strict white-list of trusted domains to match against and does not support the "null" origin.	112
14.5.4 Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.	112

Overview

This document is a byproduct of the 'CAREWare Risk Assessment Aligned with Application Security Verification Standard (ASVS) 2025' (RA) that jProg recently updated. That OWASP aligned assessment took each requirement in the ASVS and detailed what CAREWare related threats they imply and whether the jProg developers are responsible for verification of a countermeasure or whether the IT department and Administration of those running and using CAREWare servers (stakeholders) are responsible for the countermeasure. Most of the stakeholder threats will be issues most IT departments and administrators already have countermeasures for. Taking the effort to go through each threat (detailed below) and verifying your countermeasures in the 'Stakeholder status' cell, will create a Stakeholder ASVS aligned Risk Assessment that can dovetail with the larger jProg CAREWare RA thereby completing OWASP ASVS aligned RA for your CAREWare implementation.

V1 Architecture, Design and Threat Modeling

V1.1 Secure Software Development Lifecycle

1.1.1 Verify the use of a secure software development lifecycle that addresses security in all stages of development.

No stakeholder threats identified.

jProg utilises GitHub enterprise repositories and the GitKraken enterprise client for source control. For development environments we use Visual Studio .NET. for the VB .Net Business Tier and Visual Code for the golang HTTP server and javascript.

The jProg software development lifecycle includes the following:

- Direction and feedback provided through contractual negotiation with agencies that fund development and support
- jProg holds daily sprints to manage its Agile development process: We hold 2 week sprints and daily huddle.
- Detailed requirement and high-level design (HLD) documents that are reviewed by funding agencies prior implementation
- Low-level issues like how to fit the feature into the existing design and architecture are specified by a senior programmer or the chief programmer prior to time estimates or developer assignment
- The creation of test plans and regression tests are included in the development costs
- The creation of user documentation is included in the development costs
- The buildmaster assigns a target branch for the development
- Following the HLD and low-level specifications, the developer codes the feature and the build master reviews the code check-ins.
- Jenkins is used for automated builds
- Test builds are created and stakeholders are shown the feature
- Once accepted for beta testing, a build is made and the regression tester is run. Any issues found are fixed
- Beta testing involves the stakeholder installing the beta version on a test server and testing the feature with a copy of their production data
- Once the beta testing/fix/rebuild cycle is complete, the code is then deemed fully tested
- A test plan is then written for the regression tester
- A regression test that follows the test plan is then recorded and added to the CAREWare regression tester
- The regression tester is run prior to any production release
- At least twice a year, prior to these major releases, the Business Tier code is scanned for security flaws using Coverity
- At least twice a year, prior to these major releases, the Business Tier's 3rd party libraries are scanned with Black Duck

- All issues found in production releases are documented in the jProg change management system (CMS)
- Assignments to programmers are through the CMS and Agile sprint process
- Bugs and fixes are documented in the CMS and Confluence
- Bi-annual penetration tests are run by an outside firm who uses OWASP penetration testing standards and tools
- This risk assessment is reviewed and used by the pentesting firm

1.1.2 Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.

21 stakeholder threats identified.

See diagrams and notes below.

Threat Boundaries Diagram and Notes

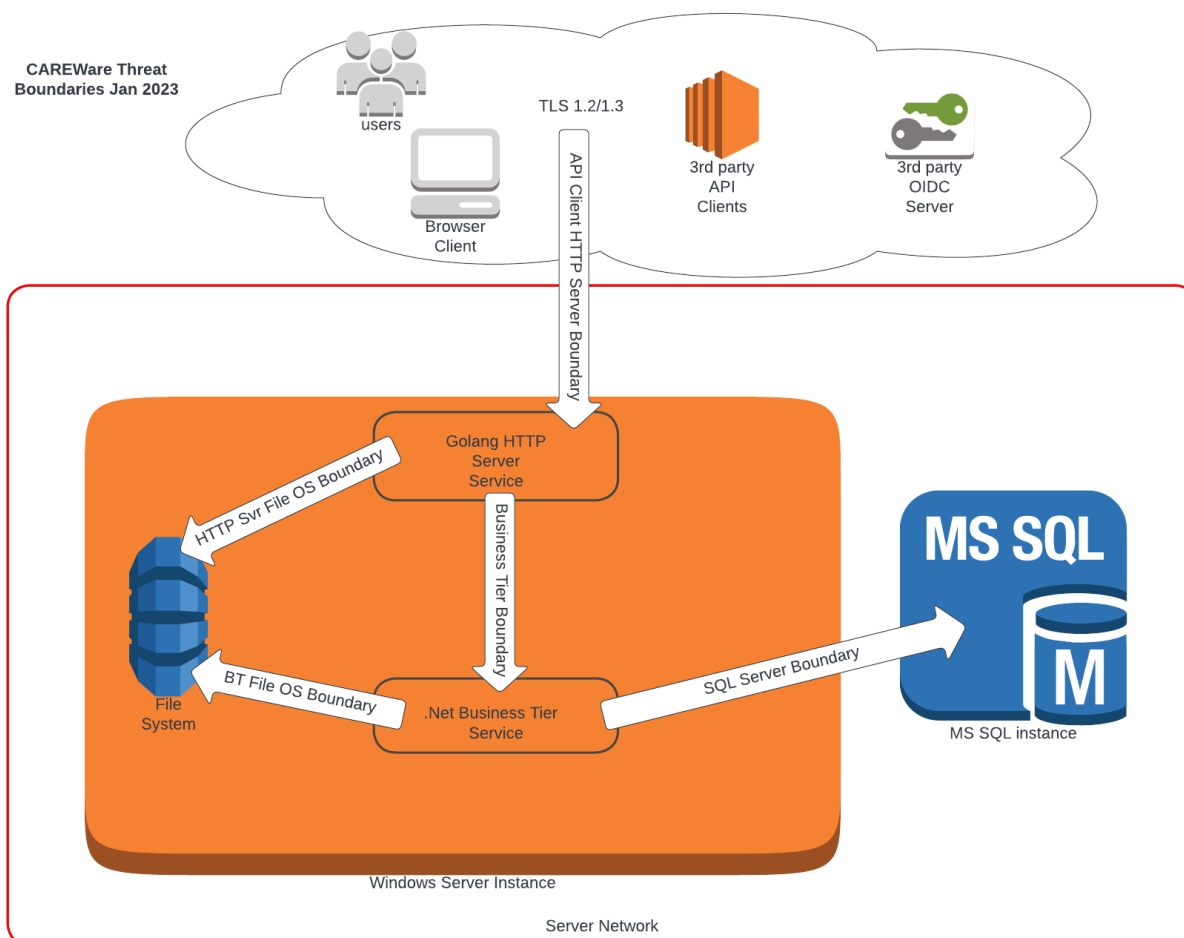


Diagram Notes

- The threat arrow tail starts with the user of the service and the head represents the responding service
- Each threat boundary has its own data flow diagram as a part of this diagram packet
- The diagram depicts the recommended configuration with the HTTP and the Business Tier services residing on the same server

Threat Identification for HTTP Server Boundary

Threat	Failure to configure TLS
Attacker	Attackers can intercept and read the cleartext HTML messages when TLS is not used.

Countermeasure	Verify CAREWare HTTP Server is configured for TLS
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Failure to secure firewall, network or HTTP server instance
Attacker	Attackers can use mishandled network and server configurations to bypass the HTTP server
Countermeasure	Verify firewalls and network and server instances have secure configurations
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat Identification for HTTP Server OS File Boundary

Threat	File System Compromised
Attacker	Attackers can access the OS File system of the HTTP server and change its configuration or steal data
Countermeasure	Verify that access to the diagramed directories is restricted
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat Identification for Business Tier Boundary

Threat	Access to localhost:8000//getdocument not restricted
Attacker	Attackers can use the access to the Business Tier to try to gain backdoor access
Countermeasure	Verify that access to the Business Tier API is restricted to only the HTTP Server
Responsibility	Stakeholder IT and Administration

Stakeholder status	
--------------------	--

Threat Identification for Business Tier OS File Boundary

Threat	File System Compromised
Attacker	Attackers can access the OS File system of the Business Tier server and change its configuration or steal data
Countermeasure	Verify that access to the diagramed directories is restricted
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat Identification for SQL Server Boundary

Threat	SQL Server connection string is stolen
Attacker	Attackers can access the OS File system of the Business Tier server and change its configuration or steal data
Countermeasure	Verify that access to BusinessTierSettings.xml and its backups is restricted and use the CW Admin tool to encrypt the connection string
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Weak password of cwbt account
Attacker	Attackers can try to access data directly by cracking the cwbt SQL user account
Countermeasure	Verify that the password used cannot be guessed or cracked
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Failure to secure SQL Server
---------------	-------------------------------------

Attacker	Attackers can exploit an unsecured SQL Server to steal or destroy data
Countermeasure	Verify that appropriate measures have been taken to secure your SQL Server
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Data Flow Diagram for HTTP Server Boundary and Notes

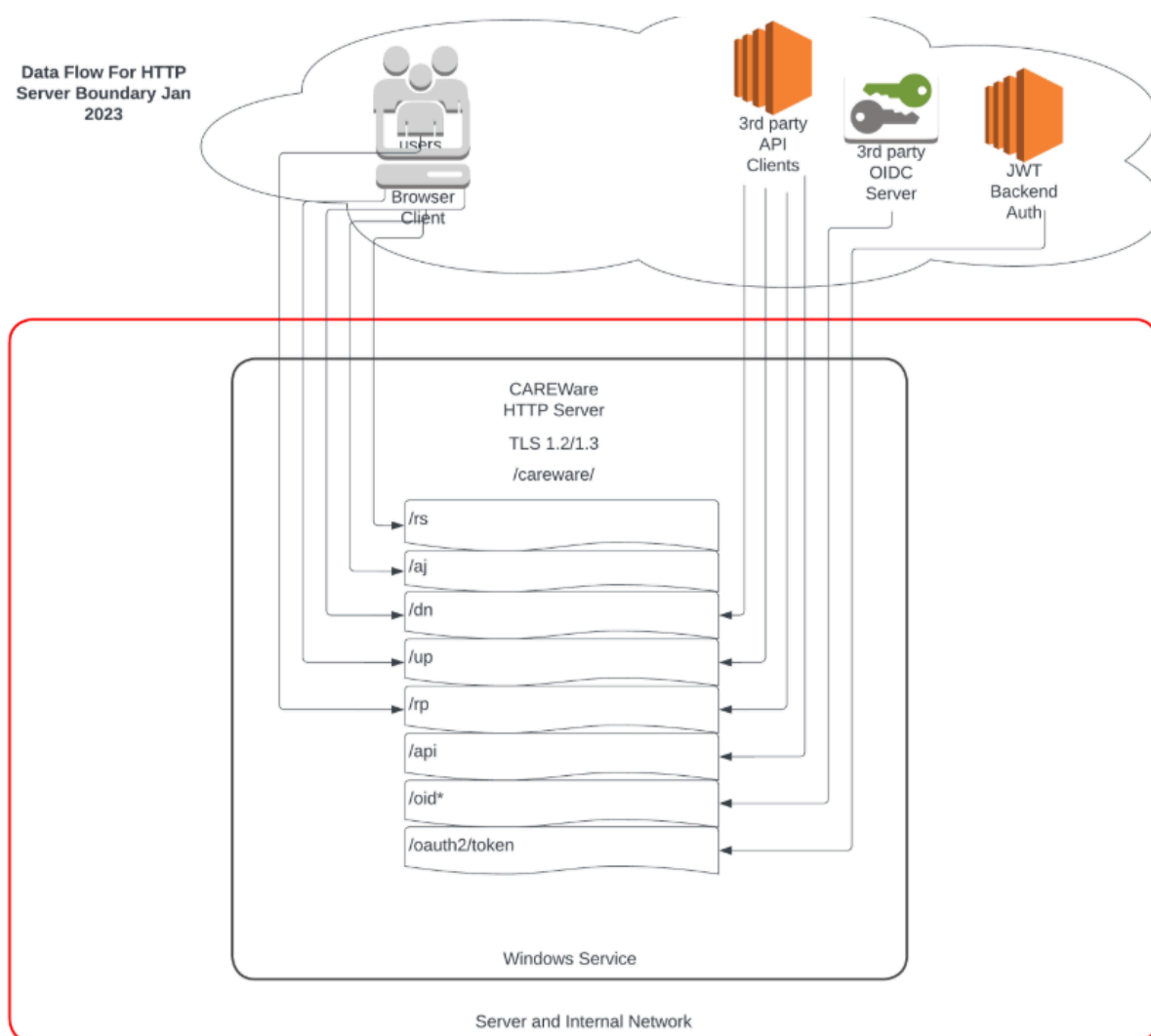


Diagram Notes

- All CAREWare API data flows through CAREWare's HTTP server, which is implemented using Google Go's HTTP Server module
- Once a successful TLS connection is established with an API client, CAREWare will only respond to HTTP requests matching the root URLs listed in the diagram
- Each of these root URLs implements a key component of CAREWare's multi-purpose API and are detailed below

CAREWare's API URL prefixes

/careware/rs

- RS is short for 'resource', these are static files like HTML, CSS and JavaScript files
- Crosses HTTP Svr File OS Boundary
- The HTTP server will map these file requests with known distributed CAREWare static files like .htm, .html and .css files
- The example path <https://example.com/careware/rs/index.htm> retrieves the static CAREWare user entry html form that is used by a browser

/careware/aj

- AJ is short for Ajax
- Crosses Business Tier Boundary
- The HTTP server processes Post requests with Ajax style JSON payloads

/careware/dn

- DN is short for file download
- Crosses Business Tier Boundary
- DN brokers the HTTP multipart download protocol with the Business Tier's 'get file chunk' API system

/careware/up

- UP is short for file upload
- Crosses Business Tier Boundary
- UP brokers the HTTP multipart upload protocol with the Business Tier's 'put file chunk' API system

/careware/rp

- RP is short for report
- Crosses Business Tier Boundary
- Crosses the HTTP Svr File OS Boundary
- RP downloads a csv file containing the row output of the report and a JSON meta file containing the report formatting instructions. It then utilizes the valyala/quicktemplate Golang module to mash the data and meta files into .html report output to the user screen

/careware/api

- Shares the same implementing function as /careware/aj
- See /careware/aj
- Available for backwards compatibility

/careware/oid*

- OID* is short for OIDC
- Implemented OIDC paths are /careware/oidreq, /careware/oidrsp, /careware/oidinfo and /careware/oidlogout

/careware/oauth2/token

- Implements OAuth2 signed JWT backend protocol

Data Flow Diagram for HTTP Server OS File Boundary

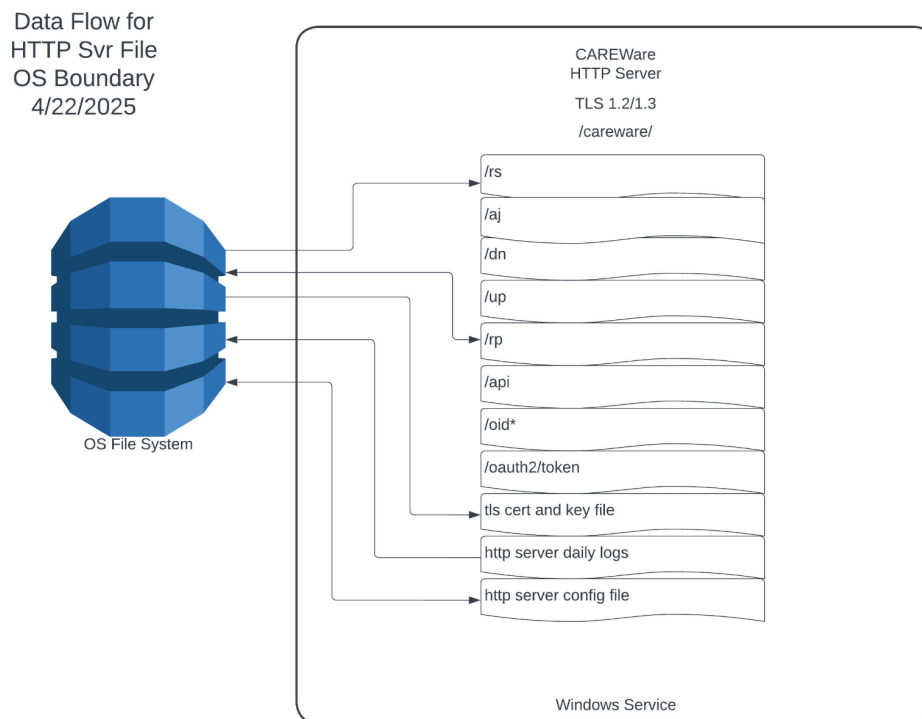


Diagram Notes

- When the HTTP server is installed, the following subdirectories are in the installation root directory

Windows (C:) > Version Control GitHub > CWJS > cwhttp >

Name	Date modified
logs	1/9/2023 3:11 PM
res_admin	1/9/2023 3:11 PM
rs	1/9/2023 2:54 PM
temprpt	1/9/2023 3:11 PM
.gitignore	1/9/2023 2:54 PM
README.md	1/9/2023 2:54 PM

/careware/rs

- RS is short for static file 'resource'
- The Google Go function behind the /rs url will only retrieve installed static files like .htm, .js and .css from the 'rs' folder where a correct path is supplied
- Threats to /rs detailed elsewhere in this report

/careware/rp

- RP is short for report
- The Google Go function behind /rp uses the 'temprpt' folder to download and then mash the requested report's underlying CSV row and JSON meta output into HTML.
- The underlying CSV and meta files are deleted once the HTML has been emitted
- Threats to /rp are detailed elsewhere in this report

TLS cert and key files

- The TLS cert and key files are placed in a directory of the choosing of IT staff, and their paths are indicated in the HTTP server's config file, which is located in the 'res_admin' folder

Threat	Failure to protect TLS cert and key files
Attacker	Attackers can use mishandled network and server configurations to bypass the HTTP server
Countermeasure	Verify Firewalls and network and server instances have secure configurations
Responsibility	Stakeholder IT and Administration
Stakeholder status	

HTTP server daily logs

- The last five days of daily log files are kept in the 'logs' folder.
- These logs can contain data about network resources and other protected system information.

Threat	Failure to protect server logs
--------	--------------------------------

Attacker	Attackers can gain valuable information about system resources to aid in further hacks
Countermeasure	Verify the server log files and their backups are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

HTTP server config file

- The server config file is a small JSON-formatted text file located in the 'res_admin' folder

Threat	Failure to protect server config file
Attacker	Attackers can gain valuable information about system resources to aid in further hacks
Countermeasure	Verify the server config file and its backups are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Data Flow Diagram for Business Tier Boundary

Data Flow for
Business Tier
Boundary
4/22/2025

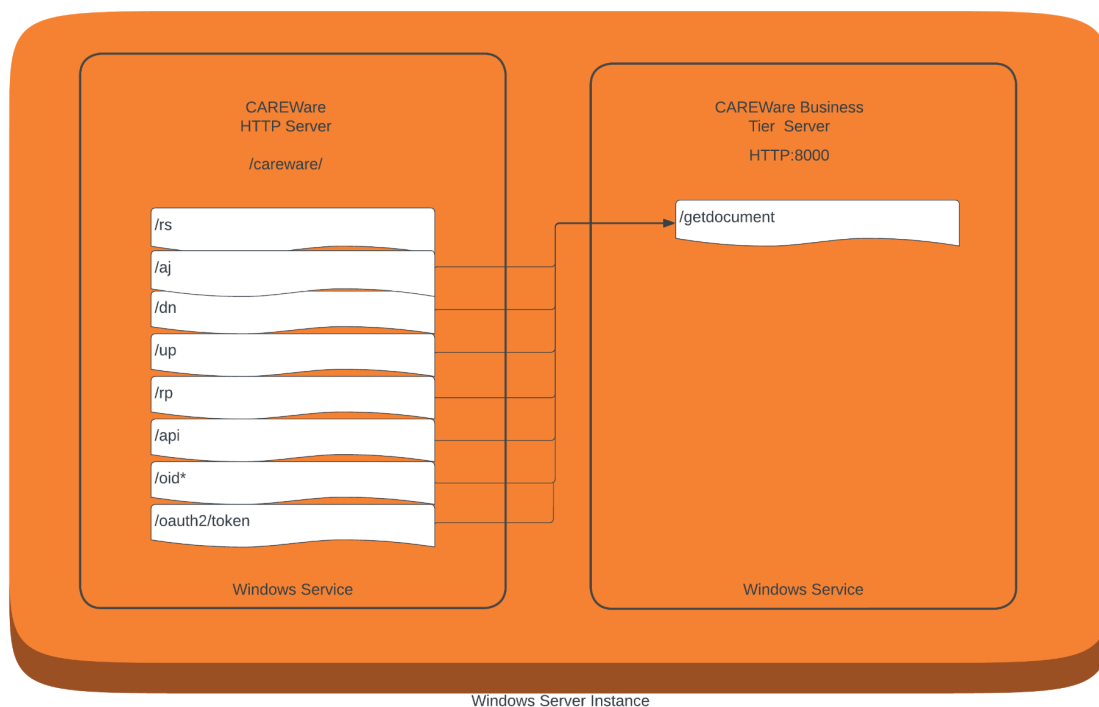


Diagram Notes

- All API requests to the Business Tier go through `/getdocument`
- The diagram depicts the recommended configuration with the HTTP and the Business Tier services residing on the same server
- Business tier boundary threats are detailed elsewhere in the document

/aj

- AJ is short for Ajax
- JSON-encoded parameters and return values

/dn

- DN is short for download
- DN brokers the HTTP multipart download protocol with the Business Tier's 'get file chunk' system used with `/getdocument`

- Each file download was previously queued for the user and assigned a random GUID that is used for a one-time retrieval of the file

/up

- UP is short for upload
- Brokers the HTTP multipart upload protocol with the Business Tier's 'put file chunk' system used with /getdocument.
- Each file upload was previously queued for the user and assigned a random GUID that is used for a one-time upload of the file

/rp

- RP is short for report
- When a report is run, the business tile creates a CSV file with the rows, columns, and a JSON-formatted meta file with the formatting instructions, and assigns them a GUID
- RP's underlying function downloads the CSV and JSON files using the Business Tier's 'get file chunk' system and then mashes them into HTML for the user

/api

- Synonym for AJ
- Uses same underlying function as /aj

/oid*

- Short for OIDC
- The underlying function uses /getdocument to retrieve stored setup information required for implementing the OIDC protocol

/oauth2/token

- The underlying function uses /getdocument to check the signature of the JWT

Data Flow Diagram for Business Tier OS File Boundary

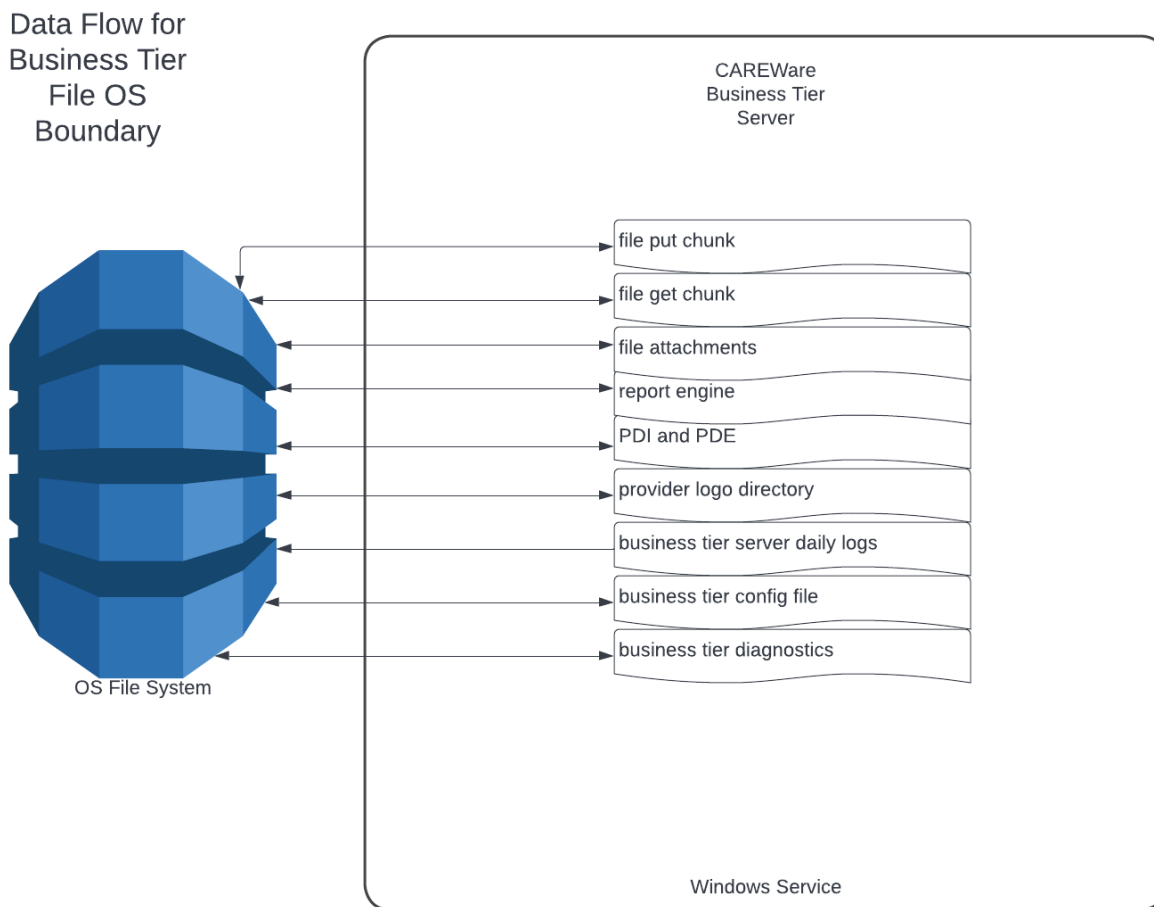


Diagram Notes

- File locations in careware are either relative to the installation directory (usually Program Files) or the location is specified by a key lookup SQL Server table called `cw_common_storage`, referred to among CW programmers and IT as 'common storage'
- See Business Tier File OS Boundary for high level threat detail

file put chunk

- When the Business Tier's 'file put chunk' document system is used, the BT creates a temporary file in the 'tempUploads' folder that gets built across calls
- Once the file is built, it is submitted to the destination process (like the PDI or file attachment system) and then deleted

file get chunk

- When the Business Tier's 'file get chunk' document system is used, the BT first retrieves the file from where it is stored and copies it to the 'tempUploads' directory and this temporary file is used to service 'file get chunk' across multiple calls
- The file is deleted from 'tempUploads' when the process is complete

Threat	tempUploads directory may contain PII and other private data
Attacker	Attackers can gain access to patient information through examining temporary files in this folder
Countermeasure	Verify that access to the tempUploads directory (and its backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

file attachments

- The Business Tier's file attachment system enables users to store supporting documents, for example to verify eligibility
- The recommended configuration is to choose a directory on a network drive that can handle the estimated usage
- The specification of the attachments directory location is required to turn the attachments feature on

Threat	file attachments directory may contain PII and other private data
Attacker	Attackers can gain access to patient information through examining files in this folder
Countermeasure	Verify that access to the file attachments directory (and its backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	user uploaded malicious file as an attachment
Attacker	Attackers can gain access to patient information through examining files in this folder
Countermeasure	Verify you have restricted file attachments to specific file extensions
Responsibility	Stakeholder IT and Administration
Stakeholder status	

report engine

- When a user runs a report in the user interface, the Business Tier outputs the column and row data to a uniquely named CSV file and directs the formatting of the report to a JSON-formatted meta file located in the 'tempUploads' directory and a retrieval key is passed to the user interface
- The Business Tier's 'file get chunk' is used by the HTTP server to retrieve these two files
- See tempUploads threat identification above

PDI and PDE

- PDI is short for Provider Data Import
- PDE is short for Provider Data Export
- The PDI/PDE is an extensive system of CSV, FHIR and HL7 import features and CSV export features that include client-level data or configuration metadata
- DTM is short for Data Translation Module, which allows users to map columns from incoming files to the CAREWare column specifications
- The PDI uses the following directories
 - PDIFolder
 - Administrators configure common storage key 'PDIWorkingDirectory' to determine where the PDI maintains the directory 'PDIFolder' and its subdirectories
 - The PDI creates directories inside of the PDIFolder. These are:
 - WorkingImports
A directory where the PDI creates or copies files to to use while completing a PDI import
 - WorkingExports
A directory where the PDI creates or copies files to to use while completing a PDE export

- AutomaticImports
A directory the PDI monitors for PDI files that are dropped in for import
- DTMIImports
A directory the DTM monitors for PDI files that are dropped in for import
- PDI_ExportDirectory is a common storage key pointing to a directory where automatic PDE export CSV files are copied to
- HL7IncomingMessageFolder is a common storage key where the BT looks for new incoming HL7-formatted files to process
- HL7ParsingErrorMessageFolder is a common storage key where the BT places incoming HL7 files for which initial processing errors occurred
- HL7ProcessedMessageFolder is a common storage key where the BT places incoming HL7 files for which the initial processing was successful
- FHIRFolder is a common storage key that the BT uses to place and look for FHIR-related documents

Threat	The PDI directories contain PII and other data
Attacker	Attackers can gain access to patient information through examining files in this folder
Countermeasure	Verify that access to the file PDI directories (and their backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

provider logo directory

- An artifact of early CAREWare development is a GUID-formatted folder created in the installation directory to hold a custom provider logo

Business Tier server daily logs

- As a default, the Business Tier maintains five days of event logs in XML format; the number of days to maintain logs is configurable
- These files are maintained in the Business Tier's installation directory

- The user interface allows administrators to view and search the contents of these files through /getdocument

Threat	The server daily logs contain sensitive information about network resources and user activity and errors
Attacker	Attackers can gain access to critical system information through examining files in this folder
Countermeasure	Verify that access to the file server daily logs (and their backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Business Tier config file

- The Business Tier reads an xml config file for startup information like the connection string to the cw_data SQL Server database.
- This file is located in the installation directory

Threat	The Business Tier config file contains sensitive SQL Server password and other restricted data
Attacker	Attackers can gain access to critical system information through examining files in this folder
Countermeasure	Verify that access to the file Business Tier config file (and its backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Installer chooses not to encrypted the SQL Server string
Attacker	Attackers can gain access to the SQL Server connection string to steal or manipulate data
Countermeasure	Use CWAdmin to Encrypt/Ofscucate SQL Server password.

Responsibility	Stakeholder IT and Administration
Stakeholder status	

Business Tier diagnostics

- Administrators can run Business Tier diagnostics sessions where all Business Tier activity is logged in JSON format and then can be reviewed by the Help Desk.
- These files are stored in the 'diagnostics' directory located in the installation directory and can be viewed using the CW_Admin utility.

Threat	The Business Tier config file contains sensitive SQL Server password and other restricted data
Attacker	Attackers can gain access to the sensitive system data
Countermeasure	Verify that access to the Business Tier diagnostics file (and its backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Data Flow Diagram for SQL Server Boundary

Data Flow for SQL Server Boundary

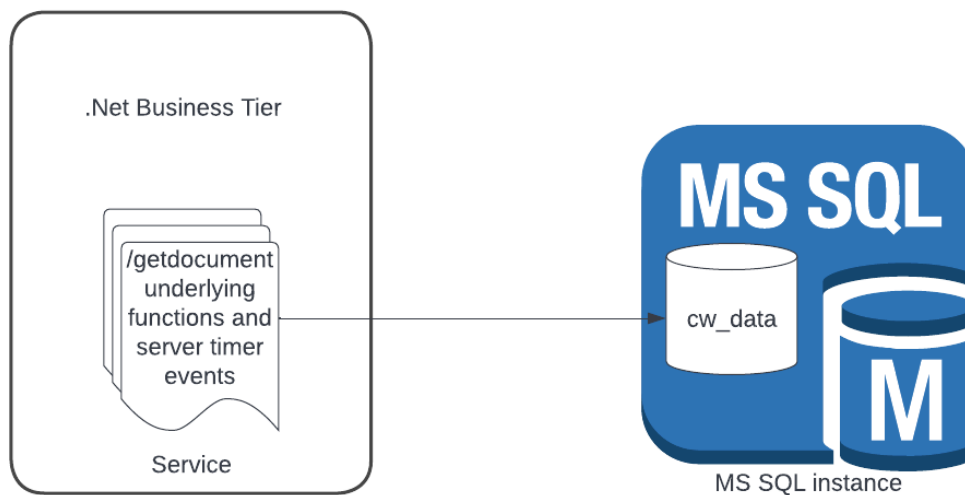


Diagram notes

- DatabaseConnectionString is the name of the key in the BT config file containing the SQL Server connection string the BT will use to try to access the cw_data database.
- An admin tool is provided for IT to set and obfuscate this connection string.
- See threats for Business Tier config file above.

Threat	Weak password for CWBT (CAREWare's SQL user)
Attacker	Attackers can gain access to the sensitive system data
Countermeasure	Verify that access to the Business Tier settings file (and its backups) are protected from attackers
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Data transmission between Business Tier and SQL server are interceptable
Attacker	Attackers can gain access to the sensitive PII and system data by intercepting SQL connection activity
Countermeasure	Verify that access to the transmission path is protected from attackers through architecture or encrypting the channel
Responsibility	Stakeholder IT and Administration
Stakeholder status	

1.1.3 Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or edit anyone else's profile"

1 stakeholder threat identified

Because CAREWare stores and fetches sensitive PII and medical data, the enforcement of user story is key to the provider protecting its information and remaining HIPAA compliant. CAREWare provides an extensive permission system that checks every request to the Business Tier to ensure the user possesses the required permissions (assigned by stakeholder administration) to access the functionality.

While CAREWare's architecture lends itself to ensuring these stories *can* be enforced, to make sure that the stories *are* enforced, jProg writes test plans that check for adherence to the story as part of the development process. Encoding this story and testing for adherence to it over time is the role of the jProg regression tester. Built into CAREWare is the ability to record the requests and replies during a user session. As part of the development process, jProg crafts a test plan and records a user session that demonstrates enforcement of the story. These recordings are played back by the regression tester, and the replies from the Business Tier are compared with the original reply. All differences in replies are logged and investigated.

Threat	Administrators do not restrict access to data on a need-to-know and need-to-modify basis
Attacker	User attackers can gain access to the sensitive PII and system data they do not need to see or modify
Countermeasure	Verify that each user only has the authorized and necessary permission
Responsibility	Stakeholder Administration

Stakeholder status	
--------------------	--

1.1.4 Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.

No stakeholder threats identified

1.1.6 Verify implementation of centralized, simple (economy of design), vetted, secure, and reusable security controls to avoid duplicate, missing, ineffective, or insecure controls.

No stakeholder threats identified

The simple transactional nature of CAREWare's Ajax Business Tier ensures secure development of new features. CAREWare's simple user interface follows a summarize, list, add, edit, and delete design pattern that forces separate BT 'documents' to be used to request each action. The API call has a document ID, and its required user permissions are bound at build time. Before any implementing function has been called, the user permissions have already been checked. If a developer forgets to associate permissions with an Ajax document, then no user will have a permission to use it, and CAREWare will throw an exception.

1.1.7 Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers.

No stakeholder threats identified

V1.2 Authentication Architecture

1.2.1 Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.

1 stakeholder threat identified

Threat	If the HTTP service or Business Tier service is compromised, CAREWare can be used to launch other attacks with system privileges
Attacker	Attackers can use compromised services to launch other attacks with system privileges
Countermeasure	Configure the HTTP Server and BT Server with least necessary privileges
Responsibility	Stakeholder IT and Administration
Stakeholder status	

1.2.2 Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed.

1 stakeholder threat identified

- See 1.1.2 Diagram and Notes.

Threat	Overbroad privileges granted components can be leveraged if compromised
Attacker	Attackers can use compromised services to launch other attacks with system privileges
Countermeasure	Configure components defined in the threat boundary diagram to have only access to what it needs
Responsibility	Stakeholder IT and Administration
Stakeholder status	

1.2.3 Verify that the application uses a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches.

No stakeholder threats identified

- CAREWare's internal authentication system has been vetted through years of pentesting and use in the field
- As a default, CAREWare locks a user account after three missed passwords; FailedLoginsBeforeLockout is configurable
- CAREWare's extensive user monitoring system records all attempted logins and their outcomes along with the source IP
- CAREWare logs all login attempts where the user name does not match along with the source IP
- CAREWare user activity monitoring system allows users to search and monitor user activity

1.2.4 Verify that all authentication pathways and identity management APIs implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application

1 stakeholder threat identified

- CAREWare can be configured to authenticate using OpenID Connect or our internal authentication system. In the case of OpenID connect, the burden of strength for authentication is on the Stakeholder
- CAREWare's authentication allows users to be designated as API users and bypass two-factor authentication. In this case, IT and Administration can control the strength of the password and its protection

Threat	IT or Administrators can create an API user with weak or compromised credentials
Attacker	Attackers can bypass 2FA by guessing an API user password can steal or compromise data
Countermeasure	Verify use of complex passwords for API users and verify that the passwords are protected

Responsibility	Stakeholder IT and Administration
Stakeholder status	

V1.3 Session Management Architecture. This is a placeholder for future architectural requirements.

This is a placeholder for future architectural requirements.

V1.4 Access Control Architecture

1.4.1 Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions enforce access controls. Never enforce access controls on the client.

No stakeholder threats identified

1.4.2 [DELETED, NOT ACTIONABLE]

1.4.3 [DELETED, DUPLICATE OF 4.1.2]

1.4.4 Verify the application uses a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths.

No stakeholder threats identified

1.4.5 Verify that attribute or feature-based access control is used whereby the code checks the user's authorization for a feature/data item rather than just their role. Permissions should still be allocated using roles.

1 stakeholder threat identified

- CAREWare has a highly granular feature-based permission system

- Groups in CAREWare are collections of these individual permissions

Threat	Non-feature based schemes lead to granting permissions that some users do not need
Attacker	Attackers can target or become users with overly broad permissions assigned
Countermeasure	Configure groups in CAREWare so that they contain only the permissions they need for a type of user. Use these groups to administer need-to-know and need-to-modify feature access
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V1.5 Input and Output Architecture

1.5.1 Verify that input and output requirements clearly define how to handle and process data based on type, content, and applicable laws, regulations, and other policy compliance.

No stakeholder threats identified

- CAREWare's use of TLS 1.2 and 1.3 is compliant with most organizations' standards and is compatible with HIPAA and other laws and regulations applicable to organizations that use CAREWare
- A user must first be added to CAREWare by an administrator, and that administrator is charged with using the Provider/User Manager to restrict the user's permissions to a need-to-know and need-to-modify basis
- Terms of use are governed by the CAREWare EULA

1.5.2 Verify that serialization is not used when communicating with untrusted clients. If this is not possible, ensure that adequate integrity controls (and possibly encryption if sensitive data is sent) are enforced to prevent deserialization attacks including object injection.

No stakeholder threats identified

- All untrusted clients are channeled through the CAREWare HTTP Server.
- See the API Client HTTP Server Boundary diagram and notes.

1.5.3 Verify that input validation is enforced on a trusted service layer.

No stakeholder threats identified

- The Business Tier service validates all input
- User feedback related to validations is provided from the Business Tier
- Pentesters check for this

1.5.4 Verify that output encoding occurs close to or by the interpreter for which it is intended.

No stakeholder threats identified

- The HTTP Server encodes the Ajax JSON request from API Clients.
- The Business tier decodes Ajax JSON it receives from the HTTP Server.

V1.6 Cryptographic Architecture

1.6.1 Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57

1 stakeholder threat identified

1.6.2 Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives.

No stakeholder threats identified

- The new CAREWare FHIR Data Source stores OAuth2 keys for the EMR connection in a SQL Server Table. Other cryptographic service consumers are listed in 1.6.1
- A keystore is under development at the time of writing this document

1.6.3 Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.

1 stakeholder threat identified

Threat	Stakeholders fail to use features to rotate keys
Attacker	Attackers can use old compromised keys to decrypt data
Countermeasure	Incorporate key rotation into your security plan
Responsibility	Stakeholders and IT
Stakeholder status	

1.6.4 Verify that the architecture treats client-side secrets--such as symmetric keys, passwords, or API tokens--as insecure and never uses them to protect or access sensitive data.

No stakeholder threats identified

- All keys, passwords and tokens are treated as insecure, and these are verified through code reviews and pentests

V1.7 Errors, Logging and Auditing Architecture

1.7.1 Verify that a common logging format and approach is used across the system.

No stakeholder threats identified

- The Business Tier uses a centralized logging system that maintains daily log files in XML format for a configurable number of days. A timer event copies the contents of these logs to a SQL Server table, and they can be viewed through the user interface
- The HTTP Server uses its own disk-based logging system based on the Google Go log module. Logs are retained for a configurable number of days

1.7.2 Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation.

1 stakeholder threat identified

- Contents of Business Tier logs are transmitted to the SQL Server and maintained for a configurable number of days (defaults to five). User interface exists for administrators to monitor and search log entries

Threat	Users and IT do not notice issues in the log
Attacker	Attackers' activities can be missed if the logs are not monitored
Countermeasure	Design policies and procedures specifying that the logs are monitored for suspicious activity
Responsibility	Stakeholder IT and administration
Stakeholder status	

V1.8 Data Protection and Privacy Architecture

1.8.1 Verify that all sensitive data is identified and classified into protection levels.

No stakeholder threats identified

Data protection levels

High	<ul style="list-style-type: none"> • Personal Identifying Information (PII) • Security data (keys, passwords) • System information (logs, private IP addresses, machine names, exception messages)
Medium	<ul style="list-style-type: none"> • Security documentation • Pentest results • Help Desk tickets • CMS tickets (bug reports)
Low	<ul style="list-style-type: none"> • User documentation • Support resources

1.8.2 Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture.

No stakeholder threats identified

Protection Requirements

High	<ul style="list-style-type: none"> • Secure network configuration • Secure SQL Server and backup configuration • Secure TLS configuration for primary boundary • Secure configuration of secondary boundaries • Secure need-to-know and need-to modify user configuration • Secure access to system exceptions • Secure storage of cryptography keys and passwords
Medium	<ul style="list-style-type: none"> • Distributed to stakeholders when requested as deemed appropriate
Low	<ul style="list-style-type: none"> • Freely distributed

	<ul style="list-style-type: none"> Secure distribution site
--	--

V1.9 Communications Architecture

1.9.1 Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers.

1 stakeholder threat identified

Threat	Communication between containers gets intercepted
Attacker	Attackers try to intercept communication between components
Countermeasure	Use the threat diagrams in this document to identify and encrypt communication between components that are in different containers, systems, sites, or cloud providers
Responsibility	Stakeholder IT and administration
Stakeholder status	

1.9.2 Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For example, application components should validate TLS certificates and chains.

1 stakeholder threat identified

Threat	Man in the middle attacks can be leveraged
Attacker	Attackers try to intercept communication between components
Countermeasure	jProg uses Golang's TLS module for outside communication
Responsibility	Stakeholder IT and administration
Stakeholder status	

V1.10 Malicious Software Architecture

1.10.1 Verify that a source code control system is in use, with procedures to ensure that check-ins are accompanied by issues or change tickets. The source code control system should have access control and identifiable users to allow traceability of any changes.

No stakeholder threats identified

jProg's source code control system includes:

- Issues raised and are tracked through the HappyFox ticketing system
- GitKraken and GitHub repositories are used to track, review, and approve or reject source changes
- Programmers and help desk staff operate from their unique user accounts
- Source code is scanned by a professional source scanner
- Assignments are managed through daily sprints

V1.11 Business Logic Architecture

1.11.1 Verify the definition and documentation of all application components in terms of the business or security functions they provide.

No stakeholder threats identified

- These components and functions are documented in this RA

1.11.2 Verify that all high-value business logic flows, including authentication, session management and access control, do not share unsynchronized state.

No stakeholder threats identified

- CAREWare currently has a single instance architecture

1.11.3 Verify that all high-value business logic flows, including authentication, session management and access control are thread safe and resistant to time-of-check and time-of-use race conditions.

No stakeholder threats identified

- Use of libraries and application architecture are designed to be thread safe and to avoid race conditions

V1.12 Secure File Upload Architecture

1.12.1 [DELETED, DUPLICATE OF 12.4.1]

1.12.2 Verify that user-uploaded files - if required to be displayed or downloaded from the application - are served by either octet stream downloads, or from an unrelated domain, such as a cloud file storage bucket. Implement a suitable Content Security Policy (CSP) to reduce the risk from XSS vectors or other attacks from the uploaded file.

No stakeholder threats identified

- All file uploads and downloads use the Google Go HTTP server file upload and download system.

V1.13 API Architecture

This is a placeholder for future architectural requirements.

V1.14 Configuration Architecture

1.14.1 Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms.

1 stakeholder threat identified

Threat	IT Staff may create an unsecure network configuration
--------	---

Attacker	Attackers can leverage holes in network configuration controls
Countermeasure	Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms. And ensure that access to underlying CAREWare components are only allowed through documented boundaries
Responsibility	Stakeholder IT and administration
Stakeholder status	

1.14.2 Verify that binary signatures, trusted connections, and verified endpoints are used to deploy binaries to remote devices.

1 stakeholder threat identified

- All CAREWare releases are digitally signed with jProg's official certificate.

Threat	IT staff can install malware disguised as a component installation files
Attacker	Attackers can supply trojan binaries that get installed
Countermeasure	Verify that binary signatures, trusted connections, and verified endpoints are used to deploy binaries to remote devices. Check that any new CAREWare binaries are signed by jProg
Responsibility	Stakeholder IT and administration
Stakeholder status	

1.14.3 Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.

No stakeholder threats identified

1.14.4 Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts.

No stakeholder threats identified

- CAREWare is not deployed through this process

1.14.5 Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization.

1 stakeholder threat identified

Threat	Failure to adequately sandbox, containerize and/or isolate at the network level
Attacker	Attackers can leverage access to one component to gain access to another
Countermeasure	Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization
Responsibility	Stakeholder IT and administration.
Stakeholder status	

1.14.6 Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets.

1 stakeholder threat identified

- None of these technologies are used

V2 Authentication

V2.1 Password Security

1 stakeholder threat identified

Threat	Passwords are not enough
Attacker	Attackers can take advantage of relying only on user passwords for access
Countermeasure	Turn on two-factor authentication
Responsibility	Stakeholder Administration and IT
Stakeholder status	

2.1.1 Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined). (C6)

1 stakeholder threat identified

Threat	Weak passwords can be exploited
Attacker	Attackers can take advantage of weak passwords to impersonate a user
Countermeasure	Verify that user-set passwords are at least 12 characters in length (after multiple spaces are combined) (C6) and turn on two-factor authentication
Responsibility	Stakeholder Administration and IT
Stakeholder status	

2.1.2 Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied. (C6)

1 stakeholder threat identified

Threat	Imposing short passwords decreases password options
Attacker	Attackers can take advantage of weak passwords to impersonate a user
Countermeasure	Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied. (C6)
Responsibility	Stakeholder Administration and IT
Stakeholder status	

2.1.3 Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space. (C6)

No stakeholder threats identified

2.1.4 Verify that any printable Unicode character, including language-neutral characters such as spaces and emojis, are permitted in passwords.

No stakeholder threats identified

2.1.6 Verify that password change functionality requires the user's current and new password.

No stakeholder threats identified

2.1.7 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. (C6)

No stakeholder threats identified

* Note that since different servers can configure different password rules, we cannot check the list ahead of time for their configuration. Rather, this list will serve as a fallback for servers with weaker configurations.

2.1.8 Verify that a password strength meter is provided to help users set a stronger password.

No stakeholder threats identified

2.1.9 Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters. (C6)

1 stakeholder threat identified

Threat	Overly specific rules can be leveraged to crack passwords
Attacker	Attackers can leverage overly specific rules
Countermeasure	These rules can be configured
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.1.10 Verify that there are no periodic credential rotation or password history requirements

No stakeholder threats identified

2.1.11 Verify that "paste" functionality, browser password helpers, and external password managers are permitted

No stakeholder threats identified

2.1.12 Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as native functionality

No stakeholder threats identified

- As of build 186 this functionality is not in CAREWare

V2.2 General Authenticator Requirements

2.2.1 Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account.

1 stakeholder threat identified

- Accounts are locked after 3 missed passwords or 2FA submissions.

Threat	Automated password cracking tools can crack user accounts by trying different passwords in quick succession
---------------	--

Attacker	Attackers use automated password cracking tools
Countermeasure	Turn on 2FA, which deters the effectiveness of password cracking
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.2.2 Verify that the use of weak authenticators (such as SMS and email) is limited to secondary verification and transaction approval and not as a replacement for more secure authentication methods. Verify that stronger methods are offered before weak methods, users are aware of the risks, or that proper measures are in place to limit the risks of account compromise.

No stakeholder threats identified

2.2.3 Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification.

No stakeholder threats identified

- As of build 186 users are notified of account changes.

2.2.4 Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates

1 stakeholder threat identified

Threat	Users can be tricked into supplying credentials through trick emails
Attacker	Attackers use phishing to steal user credentials

Countermeasure	Train users how to spot and report phishing attacks
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.2.5 Verify that where a credential service provider (CSP) and the application verifying authentication are separated, mutually authenticated TLS is in place between the two endpoints.

No stakeholder threats identified

2.2.6 Verify replay resistance through the mandated use of OTP devices, cryptographic authenticators, or lookup codes.

No stakeholder threats identified

2.2.7 Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such as a button press on a FIDO hardware key.

No stakeholder threats identified

V2.3 Authenticator Lifecycle Requirements

2.3.1 Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long term password.

No stakeholder threats identified

- As of build 186 CAREWare generates random keys for password setup and reset.

2.3.2 Verify that enrollment and use of subscriber-provided authentication devices are supported, such as a U2F or FIDO tokens.

No stakeholder threats identified

- CAREWare's internal authenticator does not support U2F or FIDO tokens.

2.3.3 Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.

No stakeholder threats identified

V2.4 Credential Storage Requirements

2.4.1 Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved oneway key derivation or password hashing function. Key derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash. (C6)

No stakeholder threats identified

2.4.2 Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored. (C6)

No stakeholder threats identified

2.4.3 Verify that if PBKDF2 is used, the iteration count SHOULD be as large as verification server performance will allow, typically at least 100,000 iterations. (C6)

No stakeholder threats identified

- Does not apply. CAREWare uses bcrypt.

2.4.4 Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, typically at least 13. (C6)

No stakeholder threats identified

- As of build 186, CAREWare uses a work factor of at least 13.

2.4.5 Verify that an additional iteration of a key derivation function is performed, using a salt value that is secret and known only to the verifier. Generate the salt value using an approved random bit generator [SP 800-90Ar1] and provide at least the minimum security strength specified in the latest revision of SP 800-131A. The secret salt value SHALL be stored separately from the hashed passwords (e.g., in a specialized device like a hardware security module).

No stakeholder threats identified

V2.5 Credential Recovery Requirements

2.5.1 Verify that a system generated initial activation or recovery secret is not sent in clear text to the user. (C6)

1 stakeholder threat identified

Threat	Clear text transmission can be intercepted
Attacker	Attackers try to intercept clear text
Countermeasure	Do not transmit these secrets via clear text
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.5.2 Verify password hints or knowledge-based authentication (so-called "secret questions") are not present.

1 stakeholder threat identified

- This optional and legacy "secret questions" feature is still available in build 186.

Threat	Secret questions can be guessed
Attacker	Attackers try to find and guess secret questions
Countermeasure	Verify this feature can be turned off administratively in CAREWare
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.5.3 Verify password credential recovery does not reveal the current password in any way. (C6)

No stakeholder threats identified

2.5.4 Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").

1 stakeholder threat identified

Threat	Default admin accounts can be targeted
Attacker	Attackers target default admin accounts
Countermeasure	Adopt IT practices specifying that these accounts are not used
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.5.5 Verify that if an authentication factor is changed or replaced, that the user is notified of this event.

No stakeholder threats identified

2.5.6 Verify forgotten password, and other recovery paths use a secure recovery mechanism, such as TOTP or other soft token, mobile push, or another offline recovery mechanism. (C6)

1 stakeholder threat identified

Threat	Password recovery presents an avenue of attack
Attacker	Attackers can use weak password recovery to hijack an account
Countermeasure	Use CAREWare's email password reset functionality
Responsibility	Stakeholder IT and Administration
Stakeholder status	

2.5.7 Verify that if OTP or multi-factor authentication factors are lost, that evidence of identity proofing is performed at the same level as during enrollment.

No stakeholder threats identified

V2.6 Look-up Secret Verifier Requirements

2.6.1 Verify that lookup secrets can be used only once.

No stakeholder threats identified

- CAREWare does not use look-up secrets

2.6.2 Verify that lookup secrets have sufficient randomness (112 bits of entropy), or if less than 112 bits of entropy, salted with a unique and random 32-bit salt and hashed with an approved one-way hash.

No stakeholder threats identified

- CAREWare does not use look-up secrets

2.6.3 Verify that lookup secrets are resistant to offline attacks, such as predictable values.

No stakeholder threats identified

- CAREWare does not use look-up secrets

V2.7 Out of Band Verifier Requirements

2.7.1 Verify that clear text out of band (NIST "restricted") authenticators, such as SMS or PSTN, are not offered by default, and stronger alternatives such as push notifications are offered first.

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

2.7.2 Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

2.7.3 Verify that the out of band verifier authentication requests, codes, or tokens are only usable once, and only for the original authentication request.

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

2.7.4 Verify that the out of band authenticator and verifier communicates over a secure independent channel.

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

2.7.5 Verify that the out of band verifier retains only a hashed version of the authentication code.

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

2.7.6 Verify that the initial authentication code is generated by a secure random number generator, containing at least 20 bits of entropy (typically a six digital random number is sufficient).

No stakeholder threats identified

- CAREWare does not use out of band (NIST "restricted") authenticators

V2.8 Single or Multi Factor One Time Verifier Requirements

2.8.1 Verify that time-based OTPs have a defined lifetime before expiring.

No stakeholder threats identified

2.8.2 Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage.

No stakeholder threats identified

2.8.3 Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.

No stakeholder threats identified

2.8.4 Verify that time-based OTP can be used only once within the validity period.

No stakeholder threats identified

2.8.5 Verify that if a time-based multi factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device.

No stakeholder threats identified

2.8.6 Verify physical single factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location.

No stakeholder threats identified

2.8.7 Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know.

No stakeholder threats identified

- CAREWare does not use biometric authenticators

V2.9 Cryptographic Software and Devices Verifier Requirements

2.9.1 Verify that cryptographic keys used in verification are stored securely and protected against disclosure, such as using a TPM or HSM, or an OS service that can use this secure storage.

No stakeholder threats identified

- CAREWare does not use Cryptographic Software and Devices (FIDO keys, etc.)

2.9.2 Verify that the challenge nonce is at least 64 bits in length, and statistically unique or unique over the lifetime of the cryptographic device.

No stakeholder threats identified

- CAREWare does not use Cryptographic Software and Devices (FIDO keys, etc.)

2.9.3 Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.

No stakeholder threats identified

- CAREWare does not use Cryptographic Software and Devices (FIDO keys, etc.)

V2.10 Service Authentication Requirements

2.10.1 Verify that integration secrets do not rely on unchanging passwords, such as API keys or shared privileged accounts.

No stakeholder threats identified

2.10.2 Verify that if passwords are required, the credentials are not a default account.

1 stakeholder threat identified

Threat	Default accounts are not compatible with need-to-know credentials
--------	---

Attacker	Attackers take advantage of unchanged passwords and API keys can be discovered and abused for long periods
Countermeasure	Avoid using default accounts like CWTEMP for routine access
Responsibility	Stakeholder IT and Administration.
Stakeholder status	

2.10.3 Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access.

No stakeholder threats identified

2.10.4 Verify passwords, integrations with databases and third-party systems, seeds and internal secrets, and API keys are managed securely and not included in the source code or stored within source code repositories. Such storage SHOULD resist offline attacks. The use of a secure software key store (L1), hardware trusted platform module (TPM), or a hardware security module (L3) is recommended for password storage.

No stakeholder threats identified

V3: Session Management Verification Requirements

V3.1 Fundamental Session Management Requirements

3.1.1 Verify the application never reveals session tokens in URL parameters or error messages

No stakeholder threats identified

V3.2 Session Binding Requirements

3.2.1 Verify the application generates a new session token on user authentication. (C6)

No stakeholder threats identified

3.2.2 Verify that session tokens possess at least 64 bits of entropy. (C6)

No stakeholder threats identified

3.2.3 Verify the application only stores session tokens in the browser using secure methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.

No stakeholder threats identified

3.2.4 Verify that session token are generated using approved cryptographic algorithms. (C6)

No stakeholder threats identified

V3.3 Session Logout and Timeout Requirements

3.3.1 Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties. (C6)

No stakeholder threats identified

3.3.2 If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period. (C6)

No stakeholder threats identified

3.3.3 Verify that the application terminates all other active sessions after a successful password change, and that this is effective across the application, federated login (if present), and any relying parties.

No stakeholder threats identified

3.3.4 Verify that users are able to view and log out of any or all currently active sessions and devices.

No stakeholder threats identified

V3.4 Cookie-based Session Management

3.4.1 Verify that cookie-based session tokens have the 'Secure' attribute set. (C6)

No stakeholder threats identified

3.4.2 Verify that cookie-based session tokens have the 'HttpOnly' attribute set. (C6)

No stakeholder threats identified

3.4.3 Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks. (C6)

No stakeholder threats identified

3.4.4 Verify that cookie-based session tokens use "__Host-" prefix (see references) to provide session cookie confidentiality.

No stakeholder threats identified

3.4.5 Verify that if the application is published under a domain name with other applications that set or use session cookies that might override or disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible. (C6)

No stakeholder threats identified

V3.5 Token-based Session Management

3.5.1 Verify the application does not treat OAuth and refresh tokens — on their own — as the presence of the subscriber and allows users to terminate trust relationships with linked applications.

No stakeholder threats identified

- CAREWare does not use OAuth Refresh tokens

3.5.2 Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.

No stakeholder threats identified

- CAREWare always uses session tokens

3.5.3 Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.

No stakeholder threats identified

- CAREWare does not use stateless session tokens

V3.6 Re-authentication from a Federation or Assertion

3.6.1 Verify that relying parties specify the maximum authentication time to CSPs and that CSPs re-authenticate the subscriber if they haven't used a session within that period.

No stakeholder threats identified

- CAREWare does not use CSPs

3.6.2 Verify that CSPs inform relying parties of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.

No stakeholder threats identified

- CAREWare does not use CSPs

V3.7 Defenses Against Session Management Exploits

3.7.1 Verify the application ensures a valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.

No stakeholder threats identified

V4: Access Control Verification Requirements

4.1 General Access Control Design

4.1.1 Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.

No stakeholder threats identified

4.1.2 Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.

No stakeholder threats identified

4.1.3 Verify that 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. (C7)

1 stakeholder threat identified

Threat	Users can abuse privileges they are not authorized for
Attacker	Attackers can be users
Countermeasure	Use CAREWare's Provider/User Manager to ensure users only have access to what they are specifically authorized for
Responsibility	Stakeholder IT and Administration
Stakeholder status	

4.1.4 Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new features until access is explicitly assigned. (C7)

1 stakeholder threat identified

Threat	Non-denying-by-default practices can bypass administrative control
Attacker	Attackers can be users
Countermeasure	Use CAREWare's Provider/User Manager to ensure users only have access to what they are specifically authorized for
Responsibility	Stakeholder IT and Administration
Stakeholder status	

4.1.5 Verify that access controls fail securely including when an exception occurs. (C10)

No stakeholder threats identified

V4.2 Operation Level Access Control

4.2.1 Verify that sensitive data and APIs are protected against direct object attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.

No stakeholder threats identified

4.2.2 Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.

No stakeholder threats identified

V4.3 Other Access Control Considerations

4.3.1 Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.

1 stakeholder threat identified

Threat	Passwords are not enough
Attacker	Attackers can take advantage of relying only on user passwords for access
Countermeasure	Turn on two-factor authentication

Responsibility	Stakeholder Administration and IT
Stakeholder status	

4.3.2 Verify that 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.

No stakeholder threats identified

4.3.3 Verify 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.

No stakeholder threats identified

V5: Validation, Sanitization and Encoding Verification Requirements

V5.1 Input Validation

5.1.1 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, or environment variables).

No stakeholder threats identified

5.1.2 Verify that frameworks protect against mass parameter assignment attacks, or that the application has countermeasures to protect against unsafe parameter assignment, such as marking fields private or similar. (C5)

No stakeholder threats identified

5.1.3 Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (whitelisting). (C5)

No stakeholder threats identified

5.1.4 Verify that 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 checking that suburb and zip/postcode match). (C5)

No stakeholder threats identified

5.1.5 Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content.

No stakeholder threats identified

V5.2 Sanitization and Sandboxing Requirements

5.2.1 Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature. (C5)

No stakeholder threats identified

5.2.2 Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.

No stakeholder threats identified

5.2.3 Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.

- The CAREWare alert scheduler does not sanitize user input.

No stakeholder threats identified

5.2.4 Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.

No stakeholder threats identified

5.2.5 Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.

No stakeholder threats identified

5.2.6 Verify that the application protects against SSRF attacks, by validating or sanitizing untrusted data or HTTP file metadata, such as filenames and URL input fields, use whitelisting of protocols, domains, paths and ports.

No stakeholder threats identified

5.2.7 Verify that the application sanitizes, disables, or sandboxes user-supplied SVG scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.

No stakeholder threats identified

5.2.8 Verify that the application sanitizes, disables, or sandboxes user-supplied scriptable or expression template language content, such as Markdown, CSS or XSL stylesheets, BBCode, or similar.

No stakeholder threats identified

V5.3 Output encoding and Injection Prevention Requirements

5.3.1 Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL Parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara). (C4)

No stakeholder threats identified

5.3.2 Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled. (C4)

No stakeholder threats identified

5.3.3 Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS. (C4)

No stakeholder threats identified

5.3.4 Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks. (C3)

No stakeholder threats identified

5.3.5 Verify that where parameterized or safer mechanisms are not present, context-specific output encoding is used to protect against injection attacks, such as the use of SQL escaping to protect against SQL injection. (C3, C4)

No stakeholder threats identified

5.3.6 Verify that the application projects against JavaScript or JSON injection attacks, including for eval attacks, remote JavaScript includes, CSP bypasses, DOM XSS, and JavaScript expression evaluation. (C4)

No stakeholder threats identified

5.3.7 Verify that the application protects against LDAP Injection vulnerabilities, or that specific security controls to prevent LDAP Injection have been implemented. (C4)

No stakeholder threats identified

5.3.8 Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding. (C4)

No stakeholder threats identified

5.3.9 Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.

No stakeholder threats identified

5.3.10 Verify that the application protects against XPath injection or XML injection attacks. (C4)

No stakeholder threats identified

V5.4 Memory, String, and Unmanaged Code Requirements

5.4.1 Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.

No stakeholder threats identified

5.4.2 Verify that format strings do not take potentially hostile input, and are constant.

No stakeholder threats identified

5.4.3 Verify that sign, range, and input validation techniques are used to prevent integer overflows.

No stakeholder threats identified

V5.5 Deserialization Prevention Requirements

5.5.1 Verify that serialized objects use integrity checks or are encrypted to prevent hostile object creation or data tampering. (C5)

No stakeholder threats identified

5.5.2 Verify that the application correctly restricts XML parsers to only use the most restrictive configuration possible and to ensure that unsafe features such as resolving external entities are disabled to prevent XXE.

No stakeholder threats identified

5.5.3 Verify that deserialization of untrusted data is avoided or is protected in both custom code and third-party libraries (such as JSON, XML and YAML parsers).

No stakeholder threats identified

5.5.4 Verify that when parsing JSON in browsers or JavaScript-based backends, `JSON.parse` is used to parse the JSON document. Do not use `eval()` to parse JSON.

No stakeholder threats identified

V6: Stored Cryptography Verification Requirements

V6.1 Data Classification

6.1.1 Verify that regulated private data is stored encrypted while at rest, such as personally identifiable information (PII), sensitive personal information, or data assessed likely to be subject to EU's GDPR.

1 stakeholder threat identified

- All PII in CAREWare is stored in the SQL Server

Threat	The application may not encrypt data at rest
Attacker	Attackers try to steal data at rest
Countermeasure	Use TDE, bitlocker, or some other drive encryption technology to encrypt underlying SQL databases and logs
Responsibility	Stakeholder IT and Administration
Stakeholder status	

6.1.2 Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.

1 stakeholder threat identified

- All PII in CAREWare is stored in the SQL Server.

Threat	The application may not encrypt data at rest
Attacker	Attackers try to steal data at rest
Countermeasure	Use TDE, bitlocker, or some other drive encryption technology to encrypt underlying SQL databases and logs
Responsibility	Stakeholder IT and Administration
Stakeholder status	

6.1.3 Verify that regulated financial data is stored encrypted while at rest, such as financial accounts, defaults or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records.

1 stakeholder threat identified

- All PII (including any regulated financial data like family income) in CAREWare is stored in the SQL Server

Threat	The application may not encrypt data at rest
Attacker	Attackers try to steal data at rest
Countermeasure	Use TDE, bitlocker, or some other drive encryption technology to encrypt underlying SQL databases and logs
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V6.2 Algorithms

6.2.1 Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.

No stakeholder threats identified

6.2.2 Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography. (C8)

No stakeholder threats identified

6.2.3 Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.

No stakeholder threats identified

- As of build 186, a review of cryptographic algorithms and configurations has been ordered.

6.2.4 Verify that random number, encryption or hashing algorithms, key lengths, rounds, ciphers or modes, can be reconfigured, upgraded, or swapped at any time, to protect against cryptographic breaks. (C8)

No stakeholder threats identified

- As of build 186, a review of cryptographic algorithms and configurations has been ordered.

6.2.6 Verify that nonces, initialization vectors, and other single use numbers must not be used more than once with a given encryption key. The method of generation must be appropriate for the algorithm being used.

No stakeholder threats identified

6.2.7 Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.

No stakeholder threats identified

6.2.8 Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information

No stakeholder threats identified

V6.3 Random Values

6.3.1 Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved cryptographically secure random number generator when these random values are intended to be not guessable by an attacker.

No stakeholder threats identified

6.3.2 Verify that random GUIDs are created using the GUID v4 algorithm, and a cryptographically-secure pseudo-random number generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable.

No stakeholder threats identified

6.3.3 Verify that random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.

No stakeholder threats identified

V6.4 Secret Management

6.4.1 Verify that a secrets management solution such as a key vault is used to securely create, store, control access to and destroy secrets. (C8)

No stakeholder threats identified

6.4.2 Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations. (C8)

No stakeholder threats identified

V7: Error Handling and Logging Verification Requirements

V7.1 Log Content Requirements

7.1.1 Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form. (C9, C10)

No stakeholder threats identified

7.1.2 Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy. (C9)

No stakeholder threats identified

7.1.3 Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures. (C5, C7)

No stakeholder threats identified

7.1.4 Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens. (C9)

No stakeholder threats identified

V7.2 Log Processing Requirements

7.2.1 Verify that all authentication decisions are logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.

No stakeholder threats identified

7.2.2 Verify that all access control decisions can be logged and all failed decisions are logged. This should include requests with relevant metadata needed for security investigations.

No stakeholder threats identified

V7.3 Log Protection Requirements

7.3.1 Verify that the application appropriately encodes user-supplied data to prevent log injection. (C9)

No stakeholder threats identified

7.3.2 Verify that all events are protected from injection when viewed in log viewing software. (C9)

No stakeholder threats identified

7.3.3 Verify that security logs are protected from unauthorized access and modification. (C9)

2 stakeholder threats identified

Threat	The stakeholder may not protect text-based logs from unauthorized access
Attacker	Attackers can use the information in text-based logs for malicious purposes.
Countermeasure	Take steps to restrict access to the HTTP Svr File OS and BT Svr File OS drives and backups to protect text-based logs (see Threat Boundaries Diagram and Notes)
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	The stakeholder may not protect logs stored in the DB from unauthorized access
Attacker	Attackers can use the information in text based logs for malicious purposes
Countermeasure	Take steps to restrict user access to the User Change Log and the System Log Viewer
Responsibility	Stakeholder IT and Administration
Stakeholder status	

7.3.4 Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis. (C9)

1 stakeholder threat identified

Threat	The stakeholder may not set the system time properly on the HTTP server, Business Tier server, or SQL Server
Attacker	Attackers benefit from difficulties in tracing time-based events.
Countermeasure	Ensure the system time on the servers running CAREWare's HTTP, Business Tier, and SQL Server are synchronized
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V7.4 Error Handling

7.4.1 Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate. (C10)

No stakeholder threats identified

7.4.2 Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions. (C10)

No stakeholder threats identified

7.4.3 Verify that a "last resort" error handler is defined which will catch all unhandled exceptions. (C10)

No stakeholder threats identified**V8: Data Protection Verification Requirements**

8.1 General Data Protection

8.1.1 Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.

1 stakeholder threat identified

Threat	Sensitive data may be held in places outside the application's control
Attacker	Attackers look for areas outside the application to steal data
Countermeasure	Verify the application protects sensitive data from being cached in server components such as load balancers and application caches
Responsibility	Stakeholder IT and Administration
Stakeholder status	

8.1.2 Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.

1 stakeholder threat identified

8.1.3 Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.

No stakeholder threats identified

8.1.4 Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.

No stakeholder threats identified

- As of build 186, CAREWare does not track numbers of requests except with diagnostics turned on.

8.1.5 Verify that regular backups of important data are performed and that test restoration of data is performed.

1 stakeholder threat identified

Threat	Application may not have a valid backup of data
Attacker	Attackers can abuse the application without notice
Countermeasure	Verify that regular backups of important data are performed and that test restoration of data is performed
Responsibility	Stakeholder IT and Administration
Stakeholder status	

8.1.6 Verify that backups are stored securely to prevent data from being stolen or corrupted.

1 stakeholder threat identified

Threat	Application backups may get stolen or corrupted
Attacker	Attackers can abuse stolen backups
Countermeasure	Verify that regular backups of important data are performed and that test restoration of data is performed
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V8.2 Client-side Data Protection

8.2.1 Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.

No stakeholder threats identified

8.2.2 Verify that 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.

No stakeholder threats identified

8.2.3 Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.

No stakeholder threats identified

V8.3 Sensitive Private Data

8.3.1 Verify that sensitive data is sent to the server in the HTTP message body or headers, and that query string parameters from any HTTP verb do not contain sensitive data.

No stakeholder threats identified

8.3.2 Verify that users have a method to remove or export their data on demand.

No stakeholder threats identified

8.3.3 Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.

1 stakeholder threat identified

Threat	Users may not comply with HIPAA and other regulations
Attacker	Users may leak or misuse personal information.

Countermeasure	Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way
Responsibility	Stakeholder IT and Administration.
Stakeholder status	

8.3.4 Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data. (C8)

1 stakeholder threat identified

Threat	Administrators may not be aware of the sensitive data they are storing
Attacker	Attackers may slip through misguided policies for malicious purposes
Countermeasure	Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data. (C8)
Responsibility	Stakeholder IT and Administration
Stakeholder status	

8.3.5 Verify accessing sensitive data is audited (without logging the sensitive data itself), if the data is collected under relevant data protection directives or where logging of access is required.

1 stakeholder threat identified

Threat	Administrators may not be aware of who is accessing sensitive data they are storing
Attacker	Attackers may slip through misguided policies for malicious purposes
Countermeasure	Verify that access to sensitive data is audited through monitoring the user change logs

Responsibility	Stakeholder Administration
Stakeholder status	

8.3.6 Verify that sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.

No stakeholder threats identified

8.3.7 Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity. (C8)

1 stakeholder threat identified

Threat	Sensitive or private information may not be encrypted
Attacker	Attackers may find unencrypted client information
Countermeasure	Ensure that SQL data and its backups are encrypted
Responsibility	Stakeholder IT and Administration
Stakeholder status	

8.3.8 Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires.

1 stakeholder threat identified

Threat	Data that is no longer used presents an unnecessary attack vector
Attacker	Attackers may take advantage of data retention attack vectors
Countermeasure	Establish and verify data retention measures

Responsibility	Stakeholder IT and Administration
Stakeholder status	

V9: Communications Verification Requirements

V9.1 Communications Security Requirements

9.1.1 Verify that secured TLS is used for all client connectivity, and does not fall back to insecure or unencrypted protocols. (C8)

1 stakeholder threat identified

Threat	TLS may not be configured
Attacker	Attackers intercept unencrypted communications
Countermeasure	Configure the CAREWare HTTP server to use TLS
Responsibility	Stakeholder IT and Administration
Stakeholder status	

9.1.2 Verify using online or up to date TLS testing tools that only strong algorithms, ciphers, and protocols are enabled, with the strongest algorithms and ciphers set as preferred.

No stakeholder threats identified

9.1.3 Verify that old versions of SSL and TLS protocols, algorithms, ciphers, and configuration are disabled, such as SSLv2, SSLv3, or TLS 1.0 and TLS 1.1. The latest version of TLS should be the preferred cipher suite.

No stakeholder threats identified

V9.2 Server Communications Security Requirements

9.2.1 Verify that connections to and from the server use trusted TLS certificates. Where internally generated or self-signed certificates are used, the server must be configured to only trust specific internal CAs and specific self-signed certificates. All others should be rejected.

1 stakeholder threat identified

Threat	Internal network communication between servers can be intercepted
Attacker	Attackers can intercept unencrypted communications
Countermeasure	Analyze this RA threat boundaries diagram and encrypt any communication that can be intercepted in your network
Responsibility	Stakeholder IT and Administration
Stakeholder status	

9.2.2 Verify that encrypted communications such as TLS is used for all inbound and outbound connections, including for management ports, monitoring, authentication, API, or web service calls, database, cloud, serverless, mainframe, external, and partner connections. The server must not fall back to insecure or unencrypted protocols.

1 stakeholder threat identified

Threat	Internal network communication between servers can be intercepted
Attacker	Attackers can intercept unencrypted communications
Countermeasure	Analyze this RA threat boundaries diagram and encrypt any communication that can be intercepted in your network
Responsibility	Stakeholder IT and Administration
Stakeholder status	

9.2.3 Verify that all encrypted connections to external systems that involve sensitive information or functions are authenticated.

1 stakeholder threat identified

Threat	Internal network communication between servers can be intercepted
Attacker	Internal attackers can access non-authenticated services
Countermeasure	Ensure the SQL User is authenticated
Responsibility	Stakeholder IT and Administration
Stakeholder status	

9.2.4 Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.

No stakeholder threats identified

9.2.5 Verify that backend TLS connection failures are logged.

No stakeholder threats identified

V10: Malicious Code Verification Requirements

V10.1 Code Integrity Controls

10.1.1 Verify that a code analysis tool is in use that can detect potentially malicious code, such as time functions, unsafe file operations and network connections.

No stakeholder threats identified

V10.2 Malicious Code Search

10.2.1 Verify that the application source code and third party libraries do not contain unauthorized phone home or data collection capabilities. Where such functionality exists, obtain the user's permission for it to operate before collecting any data.

No stakeholder threats identified

10.2.2 Verify that the application does not ask for unnecessary or excessive permissions to privacy related features or sensors, such as contacts, cameras, microphones, or location.

No stakeholder threats identified

10.2.3 Verify that the application source code and third party libraries do not contain back doors, such as hard-coded or additional undocumented accounts or keys, code obfuscation, undocumented binary blobs, rootkits, or anti-debugging, insecure debugging features, or otherwise out of date, insecure, or hidden functionality that could be used maliciously if discovered.

No stakeholder threats identified

10.2.4 Verify that the application source code and third party libraries does not contain time bombs by searching for date and time related functions.

No stakeholder threats identified

10.2.5 Verify that the application source code and third party libraries does not contain malicious code, such as salami attacks, logic bypasses, or logic bombs.

No stakeholder threats identified

10.2.6 Verify that the application source code and third party libraries do not contain Easter eggs or any other potentially unwanted functionality.

No stakeholder threats identified**V10.3 Deployed Application Integrity Controls**

10.3.1 Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.

1 stakeholder threat identified

Threat	Applications may be vulnerable to malicious updates
Attacker	Attackers can use malicious updates
Countermeasure	Verify that all CAREWare distributions are signed by jProg
Responsibility	Stakeholder IT and Administration
Stakeholder status	

10.3.2 Verify that the application employs integrity protections, such as code signing or sub-resource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.

No stakeholder threats identified

10.3.3 Verify that the application has protection from sub-domain takeovers if the application relies upon DNS entries or DNS sub-domains, such as expired domain names, out of date DNS pointers or CNAMEs, expired projects at public source code repos, or transient cloud APIs, serverless functions, or storage buckets (autogen-bucket-id.cloud.example.com) or similar. Protections can include ensuring that DNS names used by applications are regularly checked for expiry or change.

1 stakeholder threat identified

Threat	Stakeholders may misconfigure sub-domains
Attacker	Attackers exploit mismanaged sub-domains
Countermeasure	Ensure the sub-domain used for CAREWare is configured correctly
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V11: Business Logic Verification Requirements

V11.1 Business Logic Security Requirements

11.1.1 Verify the application will only process business logic flows for the same user in sequential step order and without skipping steps.

No stakeholder threats identified

11.1.2 Verify the application will only process business logic flows with all steps being processed in realistic human time, i.e. transactions are not submitted too quickly.

No stakeholder threats identified

11.1.3 Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.

No stakeholder threats identified

Threat	User permissions might not be properly restricted
Attacker	Attackers may abuse user accounts
Countermeasure	Verify each user is granted permissions based on a need to know
Responsibility	Stakeholder IT and Administration
Stakeholder status	

11.1.4 Verify the application has sufficient anti-automation controls to detect and protect against data exfiltration, excessive business logic requests, excessive file uploads or denial of service attacks.

No stakeholder threats identified

11.1.5 Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modelling or similar methodologies.

No stakeholder threats identified

11.1.6 Verify the application does not suffer from "time of check to time of use" (TOCTOU) issues or other race conditions for sensitive operations.

No stakeholder threats identified

11.1.7 Verify the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt. (C9)

No stakeholder threats identified

V12: File and Resources Verification Requirements

V12.1 File Upload Requirements

12.1.1 Verify that the application will not accept large files that could fill up storage or cause a denial of service attack.

1 stakeholder threat identified

Threat	Application may accept large files that could fill up storage or cause a denial of service attack
Attacker	Attackers can upload files that could fill up storage or cause a denial of service attack
Countermeasure	Verify that you have set the maximum upload size to a value in line with your storage capacity
Responsibility	Stakeholder IA and Administration
Stakeholder status	

12.1.2 Verify that compressed files are checked for "zip bombs" - small input files that will decompress into huge files thus exhausting file storage limits.

No stakeholder threats identified

12.1.3 Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.

No stakeholder threats identified

V12.2 File Integrity Requirements

12.2.1 Verify that files obtained from untrusted sources are validated to be of expected type based on the file's content.

1 stakeholder threat identified

Threat	Stakeholders may allow file types to be different from contents
Attacker	Attackers may abuse file uploads
Countermeasure	Verify that files obtained from untrusted sources are validated to be of expected type based on the file's content
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V12.3 File execution Requirement

12.3.4 Verify that the application protects against reflective file download (RFD) by validating or ignoring user-submitted filenames in a JSON, JSONP, or URL parameter, the response Content-Type header should be set to text/plain, and the Content-Disposition header should have a fixed filename.

No stakeholder threats identified

12.3.5 Verify that untrusted file metadata is not used directly with system API or libraries, to protect against OS command injection.

No stakeholder threats identified

12.3.6 Verify that the application does not include and execute functionality from untrusted sources, such as unverified content distribution networks, JavaScript libraries, node npm libraries, or server-side DLLs.

No stakeholder threats identified

V12.4 File Storage Requirements

12.4.1 Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions, preferably with strong validation.

1 stakeholder threat identified

Threat	Stakeholders may store uploaded files in the wrong way
Attacker	Attackers may abuse uploaded files stored in the wrong way
Countermeasure	Verify that files obtained from untrusted sources are stored outside the web root with limited permissions, preferably with strong validation
Responsibility	Stakeholder IT and Administration
Stakeholder status	

12.4.2 Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload of known malicious content.

1 stakeholder threat identified

Threat	Stakeholders may not scan uploaded files with a virus scanner
Attacker	Attackers may upload files with known viruses
Countermeasure	Verify that files from untrusted sources are not uploaded and that users are running virus protection software
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V12.5 File Download Requirements

12.5.1 Verify that the web tier is configured to serve only files with specific file extensions to prevent unintentional information and source code leakage. For example, backup files (e.g. .bak), temporary working files (e.g. .swp), compressed files (.zip, .tar.gz, etc) and other extensions commonly used by editors should be blocked unless required.

No stakeholder threats identified

12.5.2 Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.

No stakeholder threats identified

V12.6 SSRF Protection Requirements

12.6.1 Verify that the web or application server is configured with a whitelist of resources or systems to which the server can send requests or load data/files from.

No stakeholder threats identified

V13: API and Web Service Verification Requirements

V13.1 Generic Web Service Security Verification Requirements

13.1.1 Verify that all application components use the same encodings and parsers to avoid parsing attacks that exploit different URI or file parsing behavior that could be used in SSRF and RFI attacks.

No stakeholder threats identified

13.1.2 Verify that access to administration and management functions is limited to authorized administrators.

1 stakeholder threat identified

Threat	Stakeholders may give administration or management functions to non-administrators
Attacker	Attackers may get access to administrative functions through lower-level users
Countermeasure	Use CAREWare's user manager to verify only administrators have administrative permissions
Responsibility	Stakeholder IT and Administration
Stakeholder status	

13.1.3 Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.

No stakeholder threats identified

13.1.4 Verify that authorization decisions are made at both the URI, enforced by programmatic or declarative security at the controller or router, and at the resource level, enforced by model-based permissions.

2 stakeholder threats identified

Threat	Application may not enforce model-based permissions
Attacker	Attackers may get access to administrative functions through lower-level users

Countermeasure	Verify that authorization decisions are made at the URI and router security is in place
Responsibility	Stakeholder IT and Administration
Stakeholder status	

Threat	Stakeholders may not implement model-based permissions
Attacker	Attackers may get access to administrative functions through lower-level users
Countermeasure	Verify that CAREWare's permission system is used to enforce model-based permissions
Responsibility	Stakeholder IT and Administration
Stakeholder status	

13.1.5 Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).

No stakeholder threats identified

V13.2 RESTful Web Service Verification Requirements

No stakeholder threats identified

CAREWare is not a RESTful Web Service

V13.3 SOAP Web Service Verification Requirements

No stakeholder threats identified

CAREWare is not a SOAP Web Service

V13.4 GraphQL and other Web Service Data Layer Security Requirements

No stakeholder threats identified

CAREWare is not a GraphQL Web Service

V14: Configuration Verification Requirements

V14.1 Build

14.1.1 Verify that the application build and deployment processes are performed in a secure and repeatable way, such as CI / CD automation, automated configuration management, and automated deployment scripts.

No stakeholder threats identified

14.1.2 Verify that compiler flags are configured to enable all available buffer overflow protections and warnings, including stack randomization, data execution prevention, and to break the build of an unsafe pointer, memory, format string, integer, or string operations are found.

No stakeholder threats identified

14.1.3 Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.

1 stakeholder threat identified

Threat	Server configuration may not be hardened
---------------	---

Attacker	Attackers may abuse non-hardened servers
Countermeasure	Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use
Responsibility	Stakeholder IT and Administration
Stakeholder status	

14.1.4 Verify that the application, configuration, and all dependencies can be redeployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.

1 stakeholder threat identified

Threat	Stakeholder may not redeploy application using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion
Attacker	Attackers may abuse holes in the deployment process
Countermeasure	Verify that the application, configuration, and all dependencies can be redeployed using automated deployment scripts built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion
Responsibility	Stakeholder IT and Administration
Stakeholder status	

14.1.5 Verify that authorized administrators can verify the integrity of all security relevant configurations to detect tampering.

1 stakeholder threat identified

- See threat boundaries diagram for relevant configuration points.

Threat	Stakeholder may not verify the integrity of all security-relevant configurations to detect tampering
---------------	---

Attacker	Attackers may abuse holes in the security configuration
Countermeasure	Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering
Responsibility	Stakeholder IT and Administration
Stakeholder status	

V14.2 Dependency

14.2.1 Verify that all components are up to date, preferably using a dependency checker during build or compile time. (C2)

No stakeholder threats identified

14.2.2 Verify that all unneeded features, documentation, samples, configurations are removed, such as sample applications, platform documentation, and default or example users.

No stakeholder threats identified

14.2.3 Verify that if application assets, such as JavaScript libraries, CSS stylesheets or web fonts, are hosted externally on a content delivery network (CDN) or external provider, Subresource Integrity (SRI) is used to validate the integrity of the asset.

No stakeholder threats identified

14.2.4 Verify that third party components come from pre-defined, trusted and continually maintained repositories. (C2)

No stakeholder threats identified

14.2.5 Verify that an inventory catalog is maintained of all third party libraries in use. (C2)

No stakeholder threats identified

14.2.6 Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behavior into the application. (C2)

No stakeholder threats identified

V14.3 Unintended Security Disclosure Requirements

14.3.1 Verify that web or application server and framework error messages are configured to deliver user actionable, customized responses to eliminate any unintended security disclosures.

No stakeholder threats identified

14.3.2 Verify that web or application server and application framework debug modes are disabled in production to eliminate debug features, developer consoles, and unintended security disclosures.

No stakeholder threats identified

14.3.3 Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.

No stakeholder threats identified

V14.4 HTTP Security Headers Requirements

14.4.1 Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1)

No stakeholder threats identified

14.4.2 Verify that all API responses contain Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).

No stakeholder threats identified

14.4.3 Verify that a content security policy (CSPv2) is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.

No stakeholder threats identified

14.4.4 Verify that all responses contain X-Content-Type-Options: nosniff.

No stakeholder threats identified

14.4.5 Verify that HTTP Strict Transport Security headers are included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.

No stakeholder threats identified

14.4.6 Verify that a suitable "Referrer-Policy" header is included, such as "no-referrer" or "same-origin"

No stakeholder threats identified

14.4.7 Verify that a suitable X-Frame-Options or Content-Security-Policy: frame-ancestors header is in use for sites where content should not be embedded in a third-party site.

No stakeholder threats identified

V14.5 Validate HTTP Request Header Requirements

14.5.1 Verify that the application server only accepts the HTTP methods in use by the application or API, including pre-flight OPTIONS.

No stakeholder threats identified

14.5.2 Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.

No stakeholder threats identified

14.5.3 Verify that the cross-domain resource sharing (CORS) Access-Control-Allow-Origin header uses a strict white-list of trusted domains to match against and does not support the "null" origin.

No stakeholder threats identified

14.5.4 Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.

No stakeholder threats identified