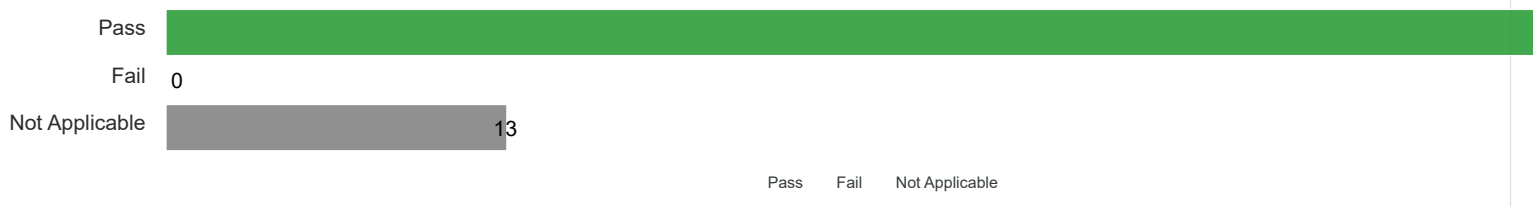
CASA Snapshot

Status

Overall CASA Status

PASS

Requirement Summary



Result Detail

Req ID	Requirement Description	CWE	Result	Comments
V1.1.4	Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.	High	✓	
V1.4.1	Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions, enforce access controls. Never enforce access controls on the client.	Medium	✓	
V1.8.1	Verify that all sensitive data is identified and classified into protection levels.	High	✓	
V1.8.2	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.	High	✓	
V1.14.6	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.	Medium	✓	Verified by Fortify Scan
V2.1.1	Verify that user set passwords are at least 12 characters in length	High	✓	Verified by Fortify Scan
V2.3.1	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.	High	Not Applicable	The developer has marked the control as NA with the following justification - "We don't use any of the initial passwords or activation codes. ". Based on which, this is marked as 'NA'.
V2.4.1	Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved one-way 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.	Low	✓	Verified by Fortify Scan
V2.5.4	Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").	Medium	✓	
V2.6.1	Verify that lookup secrets can be used only once.	High	Not Applicable	The developer has marked the control as NA with the following justification - "We don't use lookup secrets.". Based on which, this is marked as 'NA'.
V2.7.2	Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.	High	✓	
V2.7.6	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).	High	✓	
V3.3.1	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.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "It's a desktop application. We don't have any sessions. All data such as access/refresh tokens is stored on user's PC exclusively. In a secure way. Users have an option to delete the data easily in the application.". Based on which, this is marked as 'NA'.
V3.3.3	Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "It's a desktop application. We don't have any sessions. All data such as access/refresh tokens is stored on user's PC exclusively. In a secure way. Users have an option to delete the data easily in the application..". Based on which, this is marked as 'NA'.
V3.4.1	Verify that cookie-based session tokens have the 'Secure' attribute set.	Medium	✓	Verified by Fortify Scan
V3.4.2	Verify that cookie-based session tokens have the 'HttpOnly' attribute set.	Medium	✓	Verified by Fortify Scan
V3.4.3	Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks.	Medium	✓	Verified by Fortify Scan
V3.5.2	Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.	High	Not Applicable	The developer has marked the control as NA with the following justification - "It's a desktop application. We don't have any sessions. All data such as access/refresh tokens is stored on user's PC exclusively. In a secure way. Users have an option to delete the data easily in the application.". Based on which, this is marked as 'NA'.
V3.5.3	Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "It's a desktop application. We don't have any sessions. All data such as access/refresh tokens is stored on user's PC exclusively. In a secure way. Users have an option to delete the data easily in the application.". Based on which, this is marked as 'NA'.
V3.7.1	Verify the application ensures a full, valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.	High	Not Applicable	The developer has marked the control as NA with the following justification - "It's a desktop application. We don't have any sessions. All data such as access/refresh tokens is stored on user's PC exclusively. In a secure way. Users have an option to delete the data easily in the application.". Based on which, this is marked as 'NA'.
V4.1.1	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.	Medium	✓	Verified by Fortify Scan
V4.1.2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	High	✓	Verified by Fortify Scan
V4.1.3	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.	High	✓	Verified by Fortify Scan
V4.1.5	Verify that access controls fail securely including when an exception occurs.	High	✓	Verified by Fortify Scan
V4.2.1	Verify that sensitive data and APIs are protected against Insecure Direct Object Reference (IDOR) 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.	High	✓	Verified by Fortify Scan
V4.2.2	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.	High	✓	Verified by Fortify Scan
V4.3.1	Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.	Medium	✓	
V4.3.2	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.	Medium	✓	Verified by Fortify Scan
V5.1.1	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).	Medium	✓	Verified by Fortify Scan
V5.1.5	Verify that URL redirects and forwards only allow destinations which appear on an allow list, or show a warning when redirecting to potentially untrusted content.	Low		

▶	V5.2.3	Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.	Medium	✓	Verified by Fortify Scan
▶	V5.2.4	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.	Medium	✓	Verified by Fortify Scan
▶	V5.2.5	Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.	Medium	✓	Verified by Fortify Scan
▶	V5.2.6	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, and uses allow lists of protocols, domains, paths and ports.	Medium	✓	Verified by Fortify Scan
▶	V5.2.7	Verify that the application sanitizes, disables, or sandboxes user-supplied Scalable Vector Graphics (SVG) scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.	Medium	✓	
▶	V5.3.1	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 ?? or O'Hara).	High	✓	Verified by Fortify Scan
▶	V5.3.3	Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS.	High	✓	Verified by Fortify Scan
▶	V5.3.4	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.	High	✓	Verified by Fortify Scan
▶	V5.3.6	Verify that the application protects against JSON injection attacks, JSON eval attacks, and JavaScript expression evaluation.	Medium	✓	Verified by Fortify Scan
▶	V5.3.7	Verify that the application protects against LDAP injection vulnerabilities, or that specific security controls to prevent LDAP injection have been implemented.	Medium	✓	Verified by Fortify Scan
▶	V5.3.8	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.	High	✓	Verified by Fortify Scan
▶	V5.3.9	Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.	Medium	✓	Verified by Fortify Scan
▶	V5.3.10	Verify that the application protects against XPath injection or XML injection attacks.	High	✓	Verified by Fortify Scan
▶	V5.5.2	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 XML eXternal Entity (XXE) attacks.	Medium	✓	Verified by Fortify Scan
▶	V6.1.1	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.	High	Not Applicable	The developer has answered to "Does your application process or store any regulated private data, such as personally identifiable information (PII), sensitive personal information, or data assessed likely to be subject to EU's GDPR?" as "No".
▶	V6.1.2	Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.	High	Not Applicable	The developer has answered to "Does your application process or store any regulated health data, such as medical records, medical device details, or de-anonymized research records?" as "No".
▶	V6.1.3	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.<b>(High CWE)	High	Not Applicable	The developer has answered to "Does your application process or store any regulated financial data, such as financial accounts, defaults, or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records?" as "No". and also The developer has marked the control as NA with the following justification - "We don't serialize anything. ". Based on which, this is marked as 'NA'
▶	V6.2.1	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.	High	✓	Verified by Fortify Scan
▶	V6.2.3	Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.	Medium	✓	Verified by Fortify Scan
▶	V6.2.4	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.	Medium	✓	
▶	V6.2.7	Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.	Medium	✓	
▶	V6.2.8	Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information.	Medium	✓	
▶	V6.3.2	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.	Medium	✓	Verified by Fortify Scan
▶	V6.4.2	Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations.	High	✓	Verified by Fortify Scan
▶	V7.1.1	Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "We don't have direct access to user credentials or payment details. So the logging is not relevant. " and also ''We don't utilize nor log any session tokens.'' Based on which, this is marked as 'NA'.
▶	V8.1.1	Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.	Medium	✓	Verified by Fortify Scan
▶	V8.2.2	Verify that data stored in browser storage (such as localStorage, sessionStorage, IndexedDB, or cookies) does not contain sensitive data.	Medium		
▶	V8.3.1	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.	High	✓	Verified by Fortify Scan
▶	V8.3.5	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.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "We don't collect/log any user data. ". Based on which, this is marked as 'NA'.
▶	V9.1.2	Verify using up to date TLS testing tools that only strong cipher suites are enabled, with the strongest cipher suites set as preferred.	Medium	✓	Verified by Fortify Scan
▶	V9.2.1	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.	Medium	✓	Verified by Fortify Scan
▶	V9.2.4	Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.	Medium	✓	Verified by Fortify Scan
▶	V10.3.2	Verify that the application employs integrity protections, such as code signing or subresource 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.	Medium	✓	
▶	V10.3.3	Verify that the application has protection from subdomain takeovers if the application relies upon DNS entries or DNS subdomains, 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.	Medium	Not Applicable	The developer has marked the control as NA with the following justification - "The application does not rely upon DNS entries. We only communicate with Google domains that definitely cannot expire." and also ''The application does not rely upon DNS entries. We only communicate with Google domains that definitely cannot expire''. Based on which, this is marked as 'NA'.
▶	V11.1.4	Verify that the application has anti-automation controls to protect against excessive calls such as mass data exfiltration, business logic requests, file uploads or denial of service attacks.	High		
▶	V12.4.1	Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions.	Medium		

V12.4.2	Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload and serving of known malicious content.	Medium	✓	
V13.1.3	Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.	Medium		
V13.1.4	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.	High	✓	Verified by Fortify Scan
V13.2.1	Verify that enabled RESTful HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources.	High		
V14.1.1	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.	High	✓	
V14.1.4	Verify that the application, configuration, and all dependencies can be re-deployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.	High	✓	
V14.1.5	Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering.	High	✓	
V14.3.2	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.	Medium	✓	Verified by Fortify Scan
V14.5.2	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.	Medium	✓	Verified by Fortify Scan