Organization: KeePassium Labs

Application: KeePassium 1-52-151

ESOF APPSEC ADA CASA - PREMIUM REPORT

ESOF

Enterprise Security in One Framework

by TAC Security

Security is in Our DNA

1st Largest Auditor

Market share (almost 100%) of UPI assessment.

2nd Largest Financial Institution

Of countries application assessor.

3rd Largest telecom Company's

End to end security assessor.

Fortune 500

Oil and Gas company is protected by us.

10 Billion Transactions

Assessed on more than 200+ banking applications.

Top Fortune 500

Companies vulnerabilities have been managed by ESOF AppSec ADA.

Executive Summary

CASA has built upon the industry-recognized standards of the OWASP's Application Security Verification Standard (ASVS) to provide a consistent set of requirements to harden security for any application. Further, CASA provides a uniform way to perform trusted assurance assessments of these requirements when such assessments are required for applications with potential access to sensitive data.

There is no "one size fits all solution" when it comes to evaluating application risk to securing user data. The CASA assessment acknowledges this reality and is adapted with a risk-based, multi-tier assessment approach to evaluate application risk based on user, scope, and other application specific items.

Risk Classification

Critical Risk	These vulnerabilities can allow attackers to take complete control of your web applications and web servers. In exploiting this type of vulnerability, attackers could carry out a range of malicious acts.
High Risk	A High severity vulnerability, which means that on exploiting such vulnerabilities, attackers, can view information about your system that helps them find or exploit other vulnerabilities that enable them to access sensitive user and administrator information.
Medium Risk	Potential weakness in controls, which could develop into an exposure. Or Issues that represent areas of concern and may impact controls. They should be addressed reasonably promptly.
Low Risk	Potential weaknesses in controls, which in combination with other weaknesses can develop into exposure. Suggested improvements not immediately/directly affecting controls.
Info Risk	Weaknesses mentioned under these sections are informational and are best practices. Either these weaknesses cannot be exploited directly or are very difficult to exploit due to multiple constrains.

ESOF AppSec ADA Cyber Score Classification

9.7

ESOF Cyber Score

Critical	0.1-2.5
High	2.6-5.5
Medium	5.6-7.5
Low	7.6-10.0

Target

- **Title:** KeePassium 1-52-151
- **Source:**uploaded_scan_file1719857452.ipa

Testing Details

Start Date	Jul 1, 2024 18:10:52
Finish Date	Jul 2, 2024 07:40:02

SAQ

Sr. No.	Requirements	Applicable	Comments
1	Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.	Yes	Yes, app trust boudnaries, components and significant data flows are documented both in app interface and online: https://keepassium.com/articles

Sr. No.	Requirements	Applicable	Comments
2	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.	Yes	Yes, the app uses only up-to-date system libraries, without outdated technologies or plugins.
3	Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions, enforce access controls. Never enforce access controls on the client.	Yes	Yes, access to all online services is controlled on the server side by the respective service.
4	Verify that all sensitive data is identified and classified into protection levels.	Yes	Yes, all the sensitive data is protected by the app protection level (biometric scan and/or app passcode), data protection level (file encryption), memory level (encrypting most sensitive data using iOS Secure Enclave).
5	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.	Yes	Yes, each protection level has a set of associated requirements, these are applied in the app code, design and architecture.
6	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.	Yes	Application builds are signed by team-specific Apple distribution certificate. Unsigned, untrusted or tampered binaries are blocked at the system level.
7	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.	Yes	This is a standalone mobile app, there is no backend server, so the app does not rely on DNS infrastructure.
8	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.	Yes	The app runs locally on mobile device and does not have a backend server, so there is no possibility of excessive calls to that server. At the client side, automation of non-debug builds is blocked by the system. Finally, all the data in the app is owned by its user and can only be accessed by that user.
9	Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions.	No	Not a web app, does not have a backend server, there is no web root.
10	Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload and serving of known malicious content.	Yes	Files from untrusted sources are thoroughly verified for compliance with a specific format (kdb/kdbx) that cannot contain malicious content by its very structure. Thus, uploading or serving of known malicious content is impossible.
11	Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.	Yes	In all API calls (to third-party services), sensitive information is sent in encrypted HTTPS headers or request body, never in URLs. The app does not have its own backend service, so there is no risk of exposure, either.

Sr. No.	Requirements	Applicable	Comments
12	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.	Yes	Authorization decisions are enforced by programmatic security on several levels. The URI level is not applicable to our standalone mobile app that does not have a backend server.
13	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.	No	Not a web app, does not have a backend server, does not expose any API.
14	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.	Yes	Yes, build and deployment processes are scripted, secure and repeatable. Code signing configurations are managed automatically. Deployment is managed automatically on the App Store side.
15	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.	Yes	Yes, backups can be quickly restored and redeployed using the standard tools of Apple Xcode IDE.
16	Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering.	Yes	Yes, administrators can verify the integrity of all security-relevant configuration using project's code version control repository and standard tools of the Apple Xcode IDE.
17	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.	Yes	Yes, debug mode is disabled in production builds. This is additionally enforced by Apple App Store submission process, which automatically rejects debug builds.
18	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	Not a web app, does not have a backend server, does not accept HTTP requests.
19	Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks. ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering))	No	Not a web app, does not have a backend server, does not have sessions.
20	Verify that the application protects against LDAP injection vulnerabilities, or that specific security controls to prevent LDAP injection have been implemented. ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering))	No	Not a web app, does not have a backend server, does not support LDAP.
21	Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.	Yes	Yes, on several levels. The app does not load untrusted web content, runs in a system-enforced sandbox, and relies on the standard system browser which is protected against LFI and RFI.
22	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.	Yes	Yes, all the sensitive data is encrypted at rest: by iOS Secure Enclave processor, by system keychain service, and by standard encryption methods with user-defined keys.

Sr. No.	Requirements	Applicable	Comments
23	Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information.	Yes	Yes, all the cryptographic operations are implemented by well-established cryptographic libraries such as CommonCrypto, providing state-of-the-art resistance against side-channel attacks and information leaks.

Security

mail@tacsecurity.com I tacsecurity.com