Kalila Lin Cen

Security Analysis & Recommendation

Hikma Health Mobile Application

Project Description

Hikma Health is a lightweight, Android-based mobile electronic health record system designed for low-resource areas to collect and access patient health information. It offers offline access and supports multiple languages. Its intuitive design facilitates efficient patient registration and data entry in dynamic, mobile settings.

Application Outline Mechanism

The Hikma Health app combines a clean, intuitive interface with a secure backend for patient data management, supporting offline operations and multilingual capabilities in Arabic, Spanish, and English. Designed for Android devices. It offers a minimalistic and responsive design suitable for various screen sizes.

Key components include a patient section with fields for name, date of birth, sex, age, and patient summary, supporting functionalities like search, add, or edit patient information. The patient profile displays detailed information with two main buttons:

- The Trend section showing health trends over time, and 'Visit History' leading to a detailed breakdown of past visits.
- The Visit History section contains subcategories like Medical History, Allergies, Complaint, Vitals, Examination, Diagnosis, Treatment, Medicine Dispensed, Prescriptions, Notes, and COVID-19 Screening, with filter options by camp, visit type, and date. Each category leads to a page for viewing or inputting information.

Security strengths

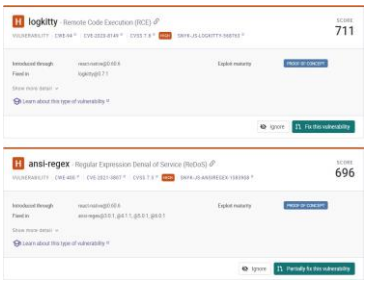
The Hikma Health app employs key security measures including SSL encryption for secure network traffic, regular updates to safeguard against vulnerabilities, secure data storage solutions to prevent unauthorized access, and careful use of updated, well-maintained third-party libraries to minimize security risks.

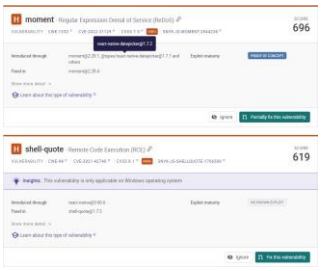
Vulnerability Analysis scanned by Snyk

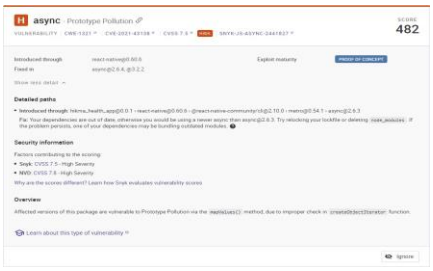
Vulnerability	Severity	Frequency	Cause	Exploitation
1.logkitty package CWE - 94 CVE-2020-8149	NVD 9.8 Critical	High	The package fails to sanitize user input before converting it into an executable command.	Attacker could potentially inject malicious code that would be executed, leading to remote code execution (RCE).
2.shell-quote package CWE -	NVD 9.8 Critical	High	The package's issue arises from mishandling shell metacharacters in a regex for Windows drive letters,	Attackers can inject unescaped metacharacters into

94 CVE-2021-42740			using the {A-z} class which includes non-standard characters like the backtick.	strings processed by the package. If used in shell commands with `exec()`, it can trigger arbitrary command execution. This Windows-specific issue relates to regex handling of drive letters.
3.async package CWE-1321 CVE-2021-43138	NVD 7.8 High	High	A defective createObjectIterator function in mapValues() leads to Prototype Pollution, affecting inherited properties across the application.	Attackers can manipulate object prototypes via crafted input to `mapValues()`, causing issues like altered object behavior, security flaws, and potential denial of service or remote code execution in applications.
4.ansi-regex package CWE - 400 CVE-2021-3807	NVD 7.5 High	High	The package has regex design issues in [[\\]()#;?]* and (?:;[-a-zA-Z\\d\\/#&.:=?%@~_]*)*.	Exploitation occurs by supplying input that triggers the inefficient regex pattern, causing excessive CPU usage or hanging and leading to a regular expression denial of service condition (ReDoS).
5.moment package CWE - 1333 CVE-2022-31129	NVD 7.5 High	High	The `preprocessRFC2822()` function in `moment`'s `from-string.js` is implemented incorrectly, causing the regex engine to take excessively long to evaluate input strings.	Exploitation occurs by inputting an extremely long string into `moment`'s date parsing functions, potentially causing high CPU usage or unresponsiveness, leading to a ReDoS.

Screenshots Evidence:







I assessed the project's security architecture quality using the CAWE Catalog, two architectural weaknesses identified are:

Security tactic	Flaws	Cause
Authorize Actors	Allocation of Resources Without Limits or Throttling	The `ansi-regex` package has flawed regex patterns that, when exploited with specific inputs, can lead to excessive CPU usage or system hangs, causing a regular expression denial of service (ReDoS). This is due to the regex patterns not being adequately limited or throttled.
Validate Inputs	Improper Input Validation	Improper Input Validation, as seen in the `.logkitty` package, occurs when it doesn't sanitize user input, allowing attackers to inject code and potentially cause remote code execution (RCE).

Countermeasures Recommendation

No.	Vulnerability	Recommendation/Solution
1	logkitty	Can be fixed by upgrading to `react-native@0.61.0`.
2	shell-quote	Fix the vulnerability by upgrading `shell-quote` to version 1.7.3 or higher. For projects on `react-native@0.60.6` with the older `shell-quote`, update to `react-native@0.62.0`.
3	async	Upgrade the async package to version 2.6.4 or 3.2.2, where the issue has been addressed.
4	ansi-regex	Update `ansi-regex` to a secure version (3.0.1, 4.1.1, 5.0.1, or 6.0.1) or upgrade `react-native` to version 0.61.0 to fix the vulnerability.
5	moment	The fix for this vulnerability is to upgrade the moment library to version 2.29.4

Security Principles Breach

Vulnerability	Explanation
1.logkitty package	● Establish Secure Defaults: Ensure default settings prevent execution of unsensitised input. ● Fail Securely: In case of suspicious input, default to a secure state by rejecting or neutralizing the input.

2.shell-quote package	● Minimise Attack Surface Area: Restrict the use of shell metacharacters and tighten regex patterns. ● Principle of Least Privilege: Limit the functions' ability to execute commands outside their necessary scope.
3.async package	● Avoid Security by Obscurity: Be transparent in handling object properties and avoid hidden manipulations. ● Defence in Depth: Apply multiple layers of checks to prevent prototype pollution, such as validating input types and contents.
4.ansi-regex package	● Keep Security Simple: Redesign regex to be efficient and less susceptible to ReDoS. ● Fail Securely: Implement measures to detect and stop processing of inputs that could trigger inefficient regex patterns.
5.moment package	● Minimise Attack Surface Area: Limit the length of acceptable input strings to prevent exploitation of the regex. ● Defence in Depth: Implement additional layers of input validation and monitoring to detect and mitigate ReDoS.

Critique legal, ethical, and privacy issues

The vulnerabilities in software packages raise serious concerns. Legally, they risk violating data protection laws like GDPR due to potential unauthorized data access and remote code execution. Ethically, these vulnerabilities show a lack of proper security measures, risking user trust and data integrity. Specially the inadequate validation of user input, inefficient resource allocation, and transparency in handling security risks.