Security Audit
CKD Calculator

(Team 33)

NephroTrack

Security audit for NephroTrack

Prepared by: Ben Hunter

Signed off by: Joe Coffee

Date: 10th/03/2025

Introduction

NephroTrack is an advanced mobile application designed to assist both clinicians and patients in monitoring the progression and risk level of chronic kidney disease (CKD). The platform features an integrated eGFR (Estimated Glomerular Filtration Rate) calculator, which allows users to accurately determine CKD stages based on clinical input. The application is built in unity and is integrated with a live Oracle Apex database.

Security concerns

Internal

1) **Weak API Authentication – If API calls from UNITY to Oracle APEX are not configured properly and lack necessary security protocols then unauthorised users may be able to access patient data from the database.**
2) **Data Transmission – If API requests are not encrypted (HTTPS) patient data could be intercepted.**
3) **Input Validation – Without proper input validation, the system may accept invalid or malicious data or SQL injections through input fields.**
4) **RBAC – If RBAC are not implemented, patients may be able to access clinician only features and may be able to retrieve sensitive data.**
5) **Plaintext Data Exposure – If sensitive personally identifiable information are transmitted or stored in plaintext they could be exposed in transmission or through oracle being compromised.**
6) **Publicly accessible variables in code – Variables containing sensitive data pulled from oracle can be exploited if left public.**

External

1) **SQL injection in oracle apex API – malicious SQL queries could allow unauthorised data access.**
2) **MITM – Without HTTPS attackers could intercept API communications containing sensitive data**
3) **Brute Force Attacks – Without rate limiting, attackers could brute force their way into gaining access through login input fields**
4) **DOS attacks – Attackers may generate large scale API or HTTPS requests to bottleneck the system and make the application unresponsive.**

5) **Phishing or social engineering – Fraudulent emails asking for password resets to trick users into revealing credentials.**

ID	Security issue	impact	Detection	Mitigation/contingency
S001	Weak API Authentication	Unauthorised access to sensitive data.	Access logs to monitor user activity.	API tokens should be implemented to authenticate access to the database.
S002	Data Transmission	Data in transit that has sensitive content could be intercepted.	Burp suite can be used monitor API requests and verify HTTPS encryption.	Use HTTPS requests to transmit data in an encrypted format.
S003	Poor Input Validation	Malicious data or SQL injection may be accepted into input fields.	Testing common SQL injection queries to ensure it doesn't execute in the database.	Input validation configured to ensure data inputted into the application is of a suitable format that restricts certain characters or SQL commands such as (; ‘ “ DROP, SELECT, INSERT).
S004	Lack of RBAC	Patients could access clinician features such as reports and patient folders etc.	Role based testing logged in as a patient to ensure clinician tools are separate.	RBAC implemented in UI and code to ensure only clinicians can access clinician features such as the batch upload tool used to import csv files and do batch calculations rather than one at a time.
S005	Plaintext Data Exposure	Intercepted, leaked or compromised data could expose personal information.	Burp suite to monitor what data is being sent/received between unity and the database and assess whether it's in plaintext and use database queries to see what results are returned.	Oracle uses TDE to encrypt data at rest, hashing can also be implemented to transform all data into an unreadable format so it's in ciphertext if intercepted or compromised. SHA256 in oracle and unity for password storage and verification.
S006	Code Vulnerabilities	Exposed patient/clinician in code.	Code review by accessing variables and functions outside of where they’re defined that may expose data.	Change access specifier to private thereby encapsulating variables and abstracting public functions that expose sensitive data to only return basic information.

				This vulnerability was addressed in Task 5 via code refactoring – specifically, changing public variables to private and encapsulating access within secure getter methods.
S007	SQL Injection	SQL injection in oracle apex could provide unauthorised access through malicious queries.	Test input fields and API behaviour.	Use bind variables and enforce input field validation.
S008	MITM Attacks	Attackers could intercept network and API traffic revealing sensitive data.	Monitor traffic using Burp Suite or Wireshark to inspect packets for signs of unencrypted transmission or tampering. Logs can also be analysed for unusual IP activity, malformed headers, or dropped connections.	Ensure that the application calls data form the database using HTTPS request for secure transmission.
S009	Brute Force Attacks	Unauthorised access if attackers can have repeated login attempts to guess credentials.	Monitor all login behaviour including attempts and failures.	Account lockout and rate limiting mechanisms implemented to lockout suspicious users who have had repeated login failure attempts.
S010	Denial of Service attacks	System could become bottlenecked and unresponsive.	Incident reports and error messages.	IP based rate limits to prevent a large volume of HTTPS requests from a specific location.
S011	Phishing Attacks	Users may be tricked into revealing credentials that provide someone with unauthorised access using a legitimate account.	User feedback and incident reports.	Educate users and enable a secure password reset process that is only accessible when a user clicks on forgot password.

Monitoring

This section outlines how we can review NephroTrack to ensure it complies with GDPR regulations and upholds the necessary security processes required from a mobile application.

Login Tracking – Implement log files that store successful and failed login attempts to monitor for suspicious activity. This can be implemented with a lockout system to prevent brute force attacks from repeated failed attempts.

Network monitoring - Using tools like Burp Suite we could intercept and inspect network API traffic between unity and the oracle apex to ensure HTTPS is functioning correctly and nothing is transmitted in plaintext.

API response testing - Monitor API responses to ensure only requested data is returned from oracle to unity and assess that only specific users such as clinicians can access certain information.

SQL and Input field validation - Basic SQL injection queries will be inputted into login and form submissions to ensure that input validation fields are enforced assessing whether user input is properly handled by unity and oracle.

In accordance with GDPR Article 32, the system ensures “security of processing” by encrypting sensitive data, limiting access through RBAC, and avoiding persistent local storage unless explicitly authorised. These measures reduce the risk of data leakage and align with NHS confidentiality expectations.

Security practices were refined through an iterative development process, with key vulnerabilities re-evaluated in later iterations as part of evolving threat modelling.

Recommendations

This section outlines the planned security practices that should be implemented based on the known risks and intended features of NephroTrack.

SHA-256 Hashing for sensitive information such as passwords will ensure the database remains secure if it is compromised, or data is intercepted in transmit as information would remain confidential in an unreadable format. The user could input their password into the unity application which would hash their password using SHA256 and then compare it to the hashed password in the database which would grant access, however if the password was exposed for example, an attacker wouldn't be able to decipher what the password actually is and use it to gain access to the application.

Oracle apex TDE (Transparent data encryption) should be enabled to encrypt data at rest within the database whilst transparently decrypting it for authorised users or applications such as unity.

Implement RBAC to enforce role restrictions ensuring patients can only access their own data and only clinicians can access additional features such as the intended batch upload feature

Enforce HTTPS for API communication between the database and unity to protect against data interception and MITM attacks

Lockout system for login attempts to prevent brute force attacks from repeated login attempts.

Future deployments should include external penetration testing to validate current defences against sophisticated real-world attack vectors. This will ensure the system meets NHS-level security assurance standards.

Refactoring measures were applied iteratively during development (see Task 5) to ensure that sensitive logic was abstracted, and access modifiers were enforced.

A thorough code review after every iteration to assess vulnerabilities and ensure any variables handling user credentials, API keys or patient data should be set to private to avoid unintended exposure.

Educate users on security practices after account creation to encourage users to create secure passwords, be vigilant and aware of phishing attempts and report issues when they're encountered.