**Student Name: Weight: 10%**

**Student ID:** **Marks:** **/23**

# Project Phase 6: Security Recommendations

## Introduction

In this phase of the project, you will work individually to research and then produce a security recommendation for your group’s mobile app project. This recommendation can focus on something that is already being incorporated into the project, or something that *may* be incorporated into the project.

While some class time will be provided, you should also plan to invest time outside of class. Please see Brightspace for exact due dates and the course schedule.

Plagiarism detection software will be used on all student submissions. Please refer to SAIT’s policies and procedures on Student Academic Conduct ([A.C.3.4.3](https://www.sait.ca/assets/documents/about-sait/policies-and-procedures/academic-student/ac-3-4-3-student-academic-conduct.pdf)) for more information.

## Instructions

1. Review all instructions and the Marking Criteria section before beginning your work.
2. This is an individual assessment; each student completes and submits their own work.
3. Choose one of the mobile app security recommendations from the [Android App Security Checklist](https://github.com/muellerberndt/android_app_security_checklist) (https://github.com/muellerberndt/android\_app\_security\_checklist).
4. Create a Word document to review your group’s mobile app project through a security lens by answering the following questions:
   1. What is your security recommendation? Why did you choose it?
      1. Some recommendations may not apply to the project (i.e.: biometric authentication, password policies, etc.)
      2. If you’re unable to determine an appropriate recommendation, see your instructor for guidance
   2. Who does the recommendation benefit (end-user, developer, etc.)?
   3. If the recommendation was found somewhere other than the provided checklist, include a link to it.
   4. When would the recommendation have to be implemented (based on how serious the security risk is)?
   5. Why do you think your project needs your recommendation?
   6. How do you think your recommendation could be applied?
      1. How feasible would the implementation be?
5. Create an issue in the GitHub repo with the same contents as the Word document.
6. You are responsible for submitting your own individual work by the posted due date. Your submission should include:
   1. A Word document containing your answers to the questions in Step 4
   2. A link to the issue on GitHub

## Marking Criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Needs Improvement** | **Good** | **Excellent** | **Marks** |
| **Effort** | Incomplete or irrelevant answers to who, what, where, why, when, and how questions. (3 marks) | Clear and relevant answers to who, what, where, why, when, and how questions.  (7 marks) | Insightful and comprehensive answers to who, what, where, why, when, and how questions. (10 marks) | **/10** |
| **Relevance of Recommendation** | Recommendation does not apply to the project and/or is not feasible. (1 mark) | Recommendation is somewhat relevant to the project. (3 marks) | Recommendation is relevant to the project and is reasonably feasible. (5 marks) | **/5** |
| **Evaluation of Feasibility** | No mention of the recommendation’s feasibility or how it could be implemented. (0 marks) | Some thought given to implementation but lacking in detail and/or critical thinking. (3 marks) | Breakdown of how the recommendation would be implemented and analysis of feasibility. (5 marks) | **/5** |
| **Submission** | Word document not uploaded to Brightspace and issue not created on GitHub. (0 marks) | Word document not uploaded to Brightspace or issue not created on GitHub. (2 marks) | Word document uploaded to Brightspace and issue created on GitHub. (3 marks) | **/3** |
| **Total** | | | | **/23** |

Answers:

1. Choose one of the mobile app security recommendations from the [Android App Security Checklist](https://github.com/muellerberndt/android_app_security_checklist) (<https://github.com/muellerberndt/android_app_security_checklist>).
   1. The chosen security recommendation is the use of Android's Keystore system for storing sensitive data such as user credentials or cryptographic keys.
2. **Create a Word document to review your group’s mobile app project through a security lens by answering the following questions:**
   1. **What is your security recommendation? Why did you choose it?\**
      1. The chosen security recommendation is the use of Android's Keystore system for storing sensitive data such as user credentials or cryptographic keys. The decision to recommend the use of Android's Keystore system for storing sensitive data such as user credentials or cryptographic keys is based on several key factors:
         1. **Enhanced Security:** The Android Keystore system offers a secure storage solution that is integrated into the Android operating system. It provides an additional layer of protection by storing cryptographic keys in a container from which they cannot be extracted easily. This makes it more challenging for malicious entities to access or compromise these keys.
         2. **Data Protection:** By storing sensitive data like user credentials and cryptographic keys in the Keystore, the risk of data breaches and unauthorized access is significantly reduced. The Keystore uses encryption to ensure that this data is safe, even if the device is compromised.
         3. **Compliance with Best Practices:** Utilizing the Android Keystore system aligns with industry best practices for data security. This is crucial for maintaining user trust and adhering to regulatory requirements related to data protection and privacy.
         4. **Maintaining User Trust:** In an era where data breaches are increasingly common, using a robust system like the Android Keystore helps in building and maintaining user trust. Users are more likely to trust and engage with applications that demonstrate a commitment to protecting their personal and sensitive data.
         5. **Reduced Risk of Exploitation:** Storing sensitive information in a less secure manner can make the application vulnerable to various types of attacks. The Keystore system mitigates this risk by providing a hardened security environment for key management.
         6. **App Reputation and Marketability:** Applications known for robust security measures, particularly those handling sensitive user data, are more likely to be favorably received in the market. This can lead to greater adoption rates and a positive reputation in the competitive app marketplace.
         7. **Future-proofing the Application:** As security threats evolve, having a system in place that is designed specifically for secure storage and management of cryptographic keys ensures that the application remains resilient against emerging security challenges.
      2. **Some recommendations may not apply to the project (i.e.: biometric authentication, password policies, etc.)**
         1. The selection of Android's Keystore system for our app, Sangeet, is driven by the critical need to safeguard user data. This system is particularly suited for apps like ours that may handle user preferences and profiles. The Keystore provides a robust and efficient framework for managing sensitive information, significantly enhancing the security of stored data. It utilizes encryption to protect user credentials and cryptographic keys, and therefore, helps in preventing unauthorized access and potential data breaches. This implementation not only fortifies our app's security but also builds user trust by ensuring their data is handled with the utmost care and protection. The choice of this system aligns perfectly with our commitment to delivering a secure and reliable user experience in the increasingly digital and interconnected world of mobile applications.
      3. **If you’re unable to determine an appropriate recommendation, see your instructor for guidance**
         1. We were able to determine an appropriate recommendation.
   2. **Who does the recommendation benefit (end-user, developer, etc.)?**
      1. The use of Android's Keystore system benefits multiple stakeholders:
         1. **End-Users**: End-users gain the most direct benefit. The Keystore system secures their personal data, like preferences and profiles, ensuring privacy and safety. This enhanced security fosters trust and confidence in the app, making users more comfortable with using it for their music needs.
         2. **Developers**: From a development perspective, integrating Keystore strengthens the app's defense against data breaches and hacking attempts. This not only safeguards user data but also helps maintain the developer's reputation and trustworthiness. It's also a proactive step in adhering to data protection and privacy standards, which are increasingly important in the digital world.
         3. **App's Ecosystem and Market Reputation**: A secure app contributes positively to the broader app ecosystem. It reduces the risk of vulnerabilities that could be exploited, enhancing overall mobile device security. Additionally, a strong focus on security can set the app apart in a competitive market, potentially attracting more users who prioritize privacy and security.
      2. In summary, by implementing the Keystore system, Sangeet can ensure robust data protection, benefiting users through enhanced security and privacy, while also supporting developers in maintaining a secure, reputable, and regulation-compliant app.
   3. **If the recommendation was found somewhere other than the provided checklist, include a link to it.** 
      1. The recommendation was found in the provided checklist.
   4. **When would the recommendation have to be implemented (based on how serious the security risk is)?**
      1. The integration of the Android Keystore system in the Sangeet app is a critical security measure that needs to be addressed early in the development lifecycle. Given the serious nature of the security risks associated with handling user data, this recommendation should be implemented during the initial stages of development. Prioritizing this integration ensures that the foundational aspects of the app are secure, laying a robust groundwork for user data protection. Implementing Keystore at an early stage helps in identifying and reducing potential vulnerabilities early on, avoiding the complexities and risks associated with retrofitting security measures at a later stage. Establishing a secure environment from the beginning is essential for maintaining user trust, complying with data protection regulations, and safeguarding the app against potential security breaches.
   5. **Why do you think your project needs your recommendation?**
      1. The integration of Android's Keystore system in our Sangeet app is essential for several reasons, each underlining the need for robust data security. In any application where user preferences, profiles, or any form of personal data are involved, security becomes paramount. For instance, our app allows users to create profiles or store preferences for their music choices, this data becomes a valuable asset that must be protected. The Keystore system provides a secure environment for storing such sensitive information, using encryption to prevent unauthorized access. This is particularly vital in scenarios where personal data could be targeted by malicious entities. For example, in apps without such security measures, user data could be easily compromised, leading to privacy violations and loss of user trust. By implementing the Keystore system, we ensure that our app not only meets industry standards for data security but also provides a safe and trustworthy platform for our users to enjoy their music. This security measure is not just about protecting data; it's about building and maintaining user trust, ensuring compliance with data protection standards, and establishing a foundation of reliability and safety in our digital offering. In the modern app market, where data breaches are increasingly common, implementing such a security feature is not just an added benefit but a fundamental necessity for the success and credibility of any user-centric app like Sangeet.
   6. **How do you think your recommendation could be applied?**
      1. To effectively apply the Android Keystore system in the React Native-based Sangeet app, we can utilize the following steps.
         1. **Leveraging React Native Libraries**: By utilizing React Native libraries like react-native-keychain which provide interfaces to Android's Keystore system. This allows secure storage of sensitive data, such as user credentials and personal settings.
         2. **Data Encryption and Storage**: Implement encryption of user data before storing it in the Keystore. This ensures that sensitive information like user preferences and profiles is securely encrypted and stored, which aids in enhancing the overall data security.
         3. **Secure Authentication Process**: Integrate the Keystore with the app's authentication system. When users log in, their credentials are securely verified against the encrypted data in the Keystore, ensuring a secure authentication process.
         4. **Protecting API Keys and Configuration Data**: Instead of embedding API keys or sensitive configuration data in the JavaScript code, securely store them in the Keystore. This approach minimizes the risk of exposing critical information.
         5. **Safe Data Retrieval Mechanisms**: Ensure that the app has secure mechanisms for retrieving and using the data from the Keystore. This involves fetching the encrypted data, decrypting it securely, and then using it in the app without exposing it to security risks.
         6. **Ongoing Security Assessments**: Regularly test and update the Keystore implementation, especially after updates to React Native or other dependencies, to ensure that there are no security vulnerabilities.
         7. **Compliance and Best Practices**: Stay updated with the latest security best practices and compliance requirements, ensuring that the app's use of Keystore aligns with industry standards.
      2. **Example** **Tutorial:**
         1. **Step 1: Setting Up the Environment**
            1. Start by creating a new React Native project, or use an existing project.
         2. **Step 2: Installing Dependencies**
            1. Install the react-native-keychain library. This library will handle the interactions with the Android Keystore system.
         3. **Step 3: Implementing Keystore in Your App**
            1. Utilize the Keychain library to store and retrieve sensitive data like user credentials or cryptographic keys. This ensures that such data is not stored in plain text and is safeguarded by Android's Keystore system.
         4. **Step 4: Implementing Additional Security Features**
            1. Encrypt sensitive data before storing it. While the Keystore system secures the data, additional encryption adds another layer of security.
            2. Securely store API keys and other sensitive configuration data. (No hardcoding).
         5. **Step 5: Testing and Security Assessments**
            1. Regularly test your app, particularly the security implementations, to identify and fix any vulnerabilities.
            2. Keep up with security best practices and ensure your app's compliance with the latest standards.
         6. **Step 6: Updating Your App**
            1. Continuously update your app's security features to address new threats and maintain robust protection.
      3. **How feasible would the implementation be?**
         1. Implementing Android's Keystore system in the React Native-based Sangeet app is a feasible task, especially given the supportive ecosystem of React Native. With libraries like react-native-keychain, the process of integrating secure storage and management of sensitive data becomes more straightforward. However, it requires a clear understanding of security principles and React Native's workings. Our team's existing skills in app design and willingness to learn new skills will facilitate this process. The main challenge lies in ensuring robust security while maintaining app performance and user experience, which makes thorough testing and continuous updates essential. In general, this implementation is essential for protecting user data and maintaining trust, aligning with the evolving security standards in mobile app development.
            1. **Supportive Ecosystem of React Native:**

React Native's ecosystem is rich with libraries and tools that simplify the integration of complex systems like Android's Keystore. Libraries such as react-native-keychain abstract much of the complexity involved, making the process more straightforward and accessible.

* + - * 1. **Availability of Resources and Community Support:**

The React Native community is vast and active, offering many resources, tutorials, and forums for assistance. This community support makes it easier to troubleshoot and find solutions to potential issues during implementation.

* + - * 1. **Compatibility with Existing Skills:**

Our team already has experience with React Native and basic app development, this background will greatly aid in the integration process. Familiarity with JavaScript and the React Native framework reduces the learning curve.

* + - * 1. **Modular and Flexible Framework:**

React Native's modular nature allows for easy integration of external modules. This flexibility means that adding a security layer like the Keystore can be done without disrupting the existing app structure.

* + - * 1. **Clear Documentation and Guided Implementation:**

Many of these libraries come with comprehensive documentation and step-by-step guides, which simplifies the implementation process.

* + - * 1. **Focus on Security Without Compromising Performance:**

While ensuring robust security, React Native also maintains a strong focus on app performance and user experience. This means that integrating security measures will not necessarily compromise the app's responsiveness or speed.

* + - * 1. **Alignment with Evolving Security Standards:**

By using Android's Keystore system, our app stays aligned with evolving security standards in mobile app development, an essential aspect for user data protection and maintaining trust.