import AuthenticationServices

func performSignIn(using requests: [ASAuthorizationRequest]) {

let controller = ASAuthorizationController(authorizationRequests: requests)

controller.delegate = self

controller.presentationContextProvider = self

controller.performRequests()

}

import Security

let query: [String: Any] = [kSecClass as String: kSecClassGenericPassword,

kSecAttrAccount as String: "user@example.com",

kSecValueData as String: "password".data(using: .utf8)!]

let status = SecItemAdd(query as CFDictionary, nil)

guard status == errSecSuccess else { throw NSError(domain: "KeychainError",

code: Int(status),

userInfo: nil) }

import LocalAuthentication

let context = LAContext()

var error: NSError?

if context.canEvaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, error: &error) {

context.evaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, localizedReason: "Access your profile") { success, authenticationError in

DispatchQueue.main.async {

if success {

// User authenticated successfully

} else {

// Handle the authentication failure

}

}

}

}

func deletePost(postID: String, user: User) -> Bool {

let userPermissions = permissions(for: user.role)

guard userPermissions.canDeletePost else {

print("User does not have permission to delete posts.")

return false

}

// Proceed with deleting the post

print("Post \(postID) deleted.")

return true

}

// Usage

deletePost(postID: "12345", user: currentUser)

import Alamofire

let accessToken = getAccessToken()

let headers: HTTPHeaders = [.authorization(bearerToken: accessToken)]

AF.request("https://api.example.com/user/profile", headers: headers).response { response in

switch response.result {

case .success(let data):

// Handle successful response

case .failure(let error):

handleTokenError(error)

}

}

<key>NSAppTransportSecurity</key>

<dict>

<key>NSAllowsArbitraryLoads</key>

<false/>

</dict>

func encryptedString(\_ input: String, key: UInt8) -> String {

return String(input.utf8.map { $0 ^ key })

}

let encryptedAPIKey = encryptedString("ActualAPIKey", key: 0x55)

func isDeviceJailbroken() -> Bool {

let fileManager = FileManager.default

if fileManager.fileExists(atPath: "/Apps/Cydia.app") ||

fileManager.fileExists(atPath: "/bin/bash") ||

fileManager.fileExists(atPath: "/usr/sbin/sshd") ||

fileManager.fileExists(atPath: "/etc/apt") {

return true

}

return false

}