iOS Application Security Controls and Countermeasures

Introduction and adaptation of new technologies like Ajax, RIA and Web Services has changed the dimension of the Application Hacking in the last few years. With introduction to iOS in mid 2007, a new era has started, it has reshaped the usage of the phone. According to a recent study, 80% of mobile users will be using smart phones by the end of 2014 out of which 25% users will be on iOS devices which is a significant market share. Even though apple reviews every application before releasing it to the AppStore, developer community should review their application before submitting it to the AppStore. Following guideline provides high level control and countermeasure to implement while developing iOS application.

Authentication

1. Validate user with unique username and password
2. Complex password should be enforced. Complex password should have following control -
   - The password must be a minimum of 8 characters long
   - Maximum password age (number of days a password can be used before the system or application requires the user to change it): 90 days
   - The password has a maximum length of 14 characters for MACs and 28 characters for PCs.
   - Passwords must contain at least one character from three of the following four character groups
     - English Uppercase Characters (A-Z)
     - English Lowercase Characters (a-z)
     - Numerals (0-9)
     - Non-alphabetic characters – (symbols such as !, $, #, %)
   - The password cannot have been the same as your previous 5 passwords
   - The password cannot contain your user name or full name
   - Password violations are limited to 10 consecutive failed password attempts before the User-ID is locked. The lockout period remains in effect for 30 minutes. Only the system administrator can have the ability to reset the User-ID
3. The application should give same error messages for wrong username and password
4. Account should be locked after 10 bad attempts of an account
5. Change password feature in application should validate an old password before allowing user to change password
6. Forgot password should send one time usable link in an email to user before allowing user to reset password
7. Forgot password should not send passwords to the user in an email
8. Use Keychain to store password into client's iPhone
   a. http://log.scifihifi.com/post/55837387/simple-iphone-keychain-code

   b. http://developer.apple.com/library/ios/#samplecode/GenericKeychain/Introduction/Intro.html
9. Information stored in the Keychain should be hashed/encrypted. In case of encryption, decryption key should not be stored as part of the application. The application should send encrypted information to the server , Decryption should happen on the server before validating the information.

Authorization

1. The application should only allow user to read/delete/update data which they are authorized to, these should be achieved by maintaining proper session at server side
2. All pages in the application should validate the active session of the user before allowing user to perform any operation
3. Proper session management should be in place
4. Session cookie should be randomly generated every time user logs in to the system
5. Session cookie should expire when the user logs out or press "home" button
6. Authorization decision should be taken after validating user's access at server side
7. Proper security controls (Secure and HTTP-Only) flag should be set on session cookies

Input validation

1. All input from the client should be validated before processed
2. The application should use stored procedures to fetch data from database
3. Sensitive data i.e. social security number, driver license number, in the database should be masked and encrypted
4. Data should be validated to protect against injection attacks i.e. SQL Injection, XPATH injection, LDAP injection
5. Forget password should reset the password
6. Protect against buffer overflow. A list of some of the functions which can cause buffer overflow can be found at https://developer.apple.com/library/ios/#documentation/Security/Conceptual/SecureCodingGuide/Articles/ValidatingInput.html
7. Use key-value coding protocol for input validation
   a. http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/KeyValueCoding/Articles/KeyValueCoding.html

Error handling

1. All errors should be handled properly in the code and generic error message should be thrown back to the user with possibly custom error code.
2. Custom error pages should be shown to the client
3. All error instances should be logged in log files

4. Implementing exception handling –
    a. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Chapters/ocExceptionHandling.html#//apple_ref/doc/uid/TP30001163-CH13-SW1

Data Security

1. Proper cache control should be in place for sensitive data
2. All sensitive data (including login credentials, social security number) should be sent over an encrypted channel (SSL)
3. SSL certificate should be matched for domain name and certification authority before performing any operation in the client
4. When a user is allowed to post any data to the application, filter to validate abuse word should be in place
5. By default, application stores last screen of the user when user press “home” button. This should be properly handled and screen shot of sensitive information i.e. username and password, account information or any other private information should not be captured as a screen shot.

Client Side validation

1. If the application is using WebView control, content should be HTML Encoded before displaying it
2. CSRF token should be sent along with each request. Token needs to be unique for each request and should be validated properly on the server side before processing the request

Logging

1. All user login attempts should be logged in server side logs
2. All error conditions should be logged in server side logs
3. Log file path should be changed from default location on the web server
4. All fail data validation attempts should be logged
5. Use NSLog to log any information on the client side.
    a. http://cocoadev.com/index.pl?NSLog

Reference

Security Coding How-To's by Apple
https://developer.apple.com/library/ios/#codinghowtos/Security/_index.html#//apple_ref/doc/uid/TP40007422

Secure Coding guide -

https://developer.apple.com/library/ios/#documentation/Security/Conceptual/SecureCodingGuide/Introduction.html#//apple_ref/doc/uid/TP40002415