Cybersecurity Blog

Everything about threat intelligence, blue team, red team, pentesting, security audit, security review, testing and assessment.

Sunday, October 9, 2016

Android Application Hardening Checklist For Developers

This checklist will help android developers to harden their application during the development phase. This checklist is essential for the developers to secure their application while in development phase only. Post development if security analyst/pentester find any loophole then it becomes tedious to dive into the existing code in order to implement the security control if there is no workaround for that loophole.

- Encrypt shared preferences
  - Reference: https://github.com/scottyab/secure-preferences
- Avoid storing sensitive information withing shared preferences even if you are using encrypted shared preferences
- Encrpyt SQLite database
  - Reference: https://github.com/sqlcipher/sqlcipher
- Application must not store any sensitive information in system logs.
- Application must not store any sensitive information in SQLitedb for any kind of analystic purpose.
- No extra permission should be taken from user than the intended one.
- Implement hmacSHA256 encryption within request and response in order to avoid tampering
  - Reference: http://infosecninja.blogspot.in/2016/09/android-application-security-tamper.html

Translate Language

Search

Search

Subscribe via email

Email address... Submit

PDFmyURL

- Use proguard/dexguard for source code ofuscation.
  - https://www.guardsquare.com
  - http://proguard.sourceforge.net/
- Disallow user to run application on rooted devices and old Android versions 4.1 >=
- Implement best available solution for SSL pinning
  - 1. Using a simple HttpsURLConnection with a PinningTrustManager
  - 2. Using a simple HttpClient with a PinningTrustManager
  - 3. Work with PinningTrustManager and PinningSSLSocketFactory more directly
  - https://github.com/moxie0/AndroidPinning
- Do not export more than required activities of post login which can be directly triggered via adb by attackers.
- If require, there must be a valid intent filter
- Do not share sensitive information on SD card which is publicly available even non-rooted devices.
- If there is file upload mechanism then follow best practices for file upload on server side.
  - Upload feature testcases - http://apps.testinsane.com/mindmaps/Uploads/Upload.png
- Implement 2/3 minutes of throttling between two consecutive requests made to the server. It helps to avoid all security scanners that can be executed on your api server, backend web application.
- Do not store sensitive data in cache at client side.
- Enable local session timeout.
  - User's account must ask for login on inactivity of 10 minutes. If user's mobile device is stolen/theft then this feature becomes handy to protect user's account and data being stolen/tampered.
- Disable debug logs
- Use secure settings for cookies
- Use tokens for account numbers instead of using account numbers directly which are pertaining to users.
- Avoid simple logic which can be easily guessable by an attackers.
- Disable clipboard copy paste
- Use latest and secure 3rd party libraries
- Implement file permissions carefully.
  - Do not create files using MODE_WORLD_READABLE and MODE_WORLD_WRITABLE
- Implement content providers carefully.
- Implement best practices for webview access.
  - Disable JavaScript access.
  - Disable local file access.
- Disallow application to run post tamper detection. Log analystics on server side for further investigation.
- Implement 2FA for financial application if required.
- Validate all inputs from client side.

► May 1 (1)
► April 10 (1)
► April 3 (2)
► 2015 (4)
► 2014 (22)
► 2013 (58)

- API server must be over TLS 1.2 or later.
- Implement web services carefully and pentest it via 3rd party server.

Posted by Frogy at 10/09/2016

Labels: android, android app, android security, encryption, hardening, hmac, sha256, sqlite, ssl

No comments:

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.

PDFmyURL.com - convert URLs, web pages or even full websites to PDF online. Easy API for developers!