Developing secure Android for Work apps

Kristian Monsen
Software Engineer
Google
Android Security 2015 Year In Review

April 2016

Developing Secure Applications for Android

- Tips for developing secure Android apps
- Overview of Google Play Services for secure app development
- Introduce the Application Security Improvement Program
Security best practices
For developers of Android applications
Always use HTTPS
➔ Should always do this for all network traffic
➔ Even more important for mobile, devices are often on untrusted networks

Use Android APIs for IPC communication
➔ Services (Binder or Messenger)
➔ Intents
➔ Broadcast Receiver
New networking APIs in N developer preview

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <base-config>
    <trust-anchors>
      <certificates src="@raw/my_ca"/>
      <certificates src="system"/>
    </trust-anchors>
  </base-config>
  <domain-config usesCleartextTraffic="false">
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="2018-01-01">
      <pin digest="SHA-256">7H1pactk1Aq2Y49orF0QKurWxmmSFZhBConYcRhJ3Y==</pin>
      <!-- backup pin -->
      <pin digest="SHA-256">fwza0LRMXouZHRcBEi+4PyulDPdcf3UKgO/04c0M1oE==</pin>
    </pin-set>
  </domain-config>
  <debug-overrides>
    <trust-anchors>
      <certificates src="@raw/debug_cas"/>
    </trust-anchors>
  </debug-overrides>
</network-security-config>
```
Use internal storage provided by Android

➔ Only accessible to the current application
➔ Avoid MODE_WORLD_(WRITEABLE|READABLE)
  ◆ Not fine grained to specific applications
  ◆ Most used alternative is content provider
➔ Optional: Encrypt files with key not available on device

External storage is world writeable/readable by default

➔ Be careful as other apps can read and modify
➔ This is also true for expansion files (saved to external storage)
Don’t dynamically load code

➔ Large security risk, very difficult to get right
  ◆ External storage
  ◆ Insecure network

➔ Expansion files are in world writeable store and very unsafe

➔ Adds complexity (testing, version management etc)
Input Validation

➔ Important on all platforms
➔ External storage is world writable
➔ Many issues with native code, but also Java can be vulnerable
➔ Script injection
➔ Use well-formatted data formats and verify before using
More information

Links with security tips and best practises for Android

➔ Security tips:
http://developer.android.com/training/articles/security-tips.html

➔ Best practises:
http://developer.android.com/training/best-security.html
Play services APIs
Updating the security provider

Make sure the application is using an updated security provider

- There have been various vulnerabilities in the security providers
  - Example: OpenSSL (CVE-2014-0224)

- Does not work if the developer uses SSLCertificateSocketFact ory directly

- It takes up to 350 ms on older devices

- There is an async and synchronous method available
Device compatibility attestation

Google API to tell you the CTS compatibility of the device

1. Simple async API
2. Read the response
3. Verify the response
4. Validating the response with Google
More information

Links to play services APIs

➔ Play services:
https://developers.google.com/android/guides/overview#the_google_play_services_client_library

➔ Attestation:
https://developer.android.com/training/safetynet/index.html

➔ Updating security provider
http://developer.android.com/training/articles/security-gms-provider.html
Application Security Improvement Program

How Google can protect users and developers in Google Play
Program overview

Find vulnerabilities
- External reports
- Internal research

Scan all apps in the Play store for vulnerability

Notify developer
- Dev console
- Email to primary contact

Remediation deadline
- 90 days after notification
- No app updates or new apps with vulnerabilities after this
Example vulnerability

OpenSSL Security Advisory [05 Jun 2014]
=================================================================

SSL/TLS MITM vulnerability (CVE-2014-0224)
=================================================================

An attacker using a carefully crafted handshake can force the use of weak keying material in OpenSSL SSL/TLS clients and servers. This can be exploited by a Man-in-the-middle (MITM) attack where the attacker can decrypt and modify traffic from the attacked client and server.

The attack can only be performed between a vulnerable client *and* server. OpenSSL clients are vulnerable in all versions of OpenSSL. Servers are only known to be vulnerable in OpenSSL 1.0.1 and 1.0.2-beta1. Users of OpenSSL servers earlier than 1.0.1 are advised to upgrade as a precaution.

OpenSSL 0.9.8 SSL/TLS users (client and/or server) should upgrade to 0.9.8za.
OpenSSL 1.0.0 SSL/TLS users (client and/or server) should upgrade to 1.0.0m.
OpenSSL 1.0.1 SSL/TLS users (client and/or server) should upgrade to 1.0.1h.

Thanks to KIKUCHI Masashi (Lepidum Co. Ltd.) for discovering and researching this issue. This issue was reported to OpenSSL on 1st May 2014 via JPCERT/CC.

The fix was developed by Stephen Henson of the OpenSSL core team partly based on an original patch from KIKUCHI Masashi.
Notify Developer

Message on the Play Developer Console

Contains a link to a help article explaining more about the vulnerability
Notify developer

Email to primary contact

Vulnerability type
Remediation details
Relevant Play policy
Affected apps
Remediation deadline

Typical campaign progression, user installations
Current results

Mostly successful, but not always

15 campaigns done so far

Best result: installed apps are fixed

Worst result: installed apps are fixed, this was a warning only campaign

It strongly depends if it is a warning and if it has a deadline
Other activities

➔ Add lint warnings for Android Studio
➔ Improve APIs so apps are safe by default
Tips about vulnerabilities

If you know of any vulnerability we should scan for we are always interested

Send email to security@android.com

Report security bug:
https://source.android.com/security/overview/updates-resources.html#report-issues
Q & A
and THANK YOU for your time.

Kristian Monsen
kristian@android.com

www.SDC2016.com