# Trusted Intent Interaction

- background: Improving trust and flexibility in interactions between Android apps
- source: https://github.com/guardianproject/TrustedIntents

the approaches to implementing the trust checking

- pinning
- TOFU/POP

tokens to check

- the package ID of the app
- the signing key of the app (see android:protectionLevel)
- a Chooser with a pinned list of package names (like PixelKnot)
- the hash of the APK
  - needed to verify that the APK has not been modfied ref
  - test exploit https://gist.github.com/poliva/36b0795ab79ad6f14fd8
  - http://www.saurik.com/id/17

### When to Verify?

There are two good opportunities for verifying senders and receivers: at install and on each Intent/startActivity() interaction. Running the verify on install means that it is rarely run but means more complicated code. Running the verify on each Intent interaction means the verification is run every time, but the code should be much simpler. Since the point of this system is security, simpler code is more important than execution efficiency.

### Chained Validation Methods

Validation is based a token. If the token does not exist in a given step, then the check goes on to the next step. If the token does have an entry but it does not match, then validation stops and an error is thrown.

1. check pins
2. check TOFU/POP
3. prompt user

## Ensuring the receiver of an Intent is trusted

If data needs to be sent to another app and the receiving app must be trusted. For example, if your app uses an OpenPGP providing app, then the sender wants to be sure that the receiver is the right app.

- isIntentReceiverTrusted()
- startTrustedActivity()
- startTrustedActivityForResult()
- bindTrustedService
- onActivityResult() would then also handle validation problems and prompts
- maybe onTrustedActivityResult()?

## Ensuring the sender of an Intent is trusted

Provide a method for a receiver to verify that the sender is the right one. For example, with ChatSecure it will become possible for other apps to send data via OTRDATA. Once the user has clicked "Accept Always" then ChatSecure should remember that and verify that each incoming Intent is allowed to be processed. Another example is ChatSecure allows apps to send an Intent to create an account.

The hard part is that Android does not provide a method to tell where an Intent came from by default. The supported way of doing it is setting a value in startActivityWithResult() to check.

- isSenderTrusted(Intent intent)
- http://www.unwesen.de/2011/04/10/android-intent-sender-verification/
- getCallingActivity() works if Intent was sent using startActivityWithResult()[1]
- RecentTaskInfo might provide enough info otherwise [2]
- Both service intents and activity intents should be handled
  - Look into Binder.getCallingPid / getCallingUid
  - If all else fails, might need an auth token exchange

- is it possible to setPackage() or set the ComponentName in an Intent then send it to a different app/Activity?

## TOFU/POP callbacks

For using TOFU/POP validation rather than pinning, there needs to be some callbacks to allow the app to display the relevant UI.

- onFirstUse() - the first time a packageName is seen
- onValidateFailed() - if there is a mismatch between the packageName and the signing certificate

## validation methods on PACKAGE_ADDED/PACKAGE_REPLACED/ PACKAGE_CHANGED

- calculating the APK hash is heavy because it has to read the whole APK
- this can be done on when receiving the @Broadcast@s from the system
- then a framework handles tracking this
- only verifies signing key or hash when the app is installed/upgraded
- lightweight version of getTrustedIntent() and startTrustedActivity()
- trusts Android system entirely
- app must include changes to AndroidManifest.xml to register BroadcastReceiver

## sources of relevant code

- PackageManager.addPermission
- Intent.ACTION_PACKAGE_ADDED
- https://github.com/moxie0/AndroidPinning
- https://github.com/ge0rg/MemorizingTrustManager
- https://github.com/guardianproject/ICTD/blob/master/src/info/guardianproject/ictd/ICTD.java#L62 (InformaCam)
- martus-android's version of OrbotHelper.java
- http://www.unwesen.de/2011/04/10/android-intent-sender-verification/
- https://github.com/commonsguy/cw-omnibus/tree/master/MiscSecurity
  - The SigDump project lists all packages -- tapping on one decodes the "signature" and dumps the signature as a binary
  - The SigCheck project checks another app's "signature", comparing it to a known good value held as a raw resource (e.g., one dumped via SigDump).

## Use Cases

### Trusted Processing App

Martus needs to have strongly integrated, trusted camera app. In this case, the Martus app will always initiate the camera Activity by verifying the recipient before sending the explicit Intent. The receiver is then expected to reply with the media directly to Martus.

An app wants to include a whitelist of trusted OpenPGP provider engires. Gnu Privacy Guard and OpenKeychain are both trusted, so the app includes pins for the signing certificates for both Gnu Privacy Guard and OpenKeychain. This app will then only send data to be encrypted or decrypted to APKs that have been signed by those pinned signing certificates.

### Per-app permissions

InformaCam aims to be engine that provides verifiable media to any app. It has app-specific media stores and keys. When an app calls InformaCam the first time, InformaCam will register the sender, assign that sender a mediastore, and only send info from that mediastore to the app that is registered to it. When sending media from a given mediastore, InformaCam will verify that it is sending to the trusted recipient for that mediastore.

## related idea for upgrading an APK's signing key

1. create an APK with new package ID and signed by new key
2. both old and new APKs are installed at the same time
3. old one grants the new one perms to read all data based on hash pin