For your updated list of attacks, you are required to
(1) find a way to detect instances of those attacks, and
(2) determine how those attacks may be prevented.
In your explanations, you should rely on the specific design decisions in the Android OS that made this attack possible (i.e., the underlying architectural causes of each attack). You should be able to find most, but possibly not all, of the information you need. For example, one or more of the papers provided to you may describe ways of detecting a given attack once it has occurred, but not ways of preventing it from happening in the first place.

Activity hijacking (Unauthorized intent receipt)

- Legitimate app needs to send an email, has implicit intent for app to do that action, user chooses malicious app instead of legit
    - "The malicious application can provide a useful service so that the user willingly makes it the default application to launch. For example, a user might opt to make a malicious browser the default browser and never get prompted to choose between components again. "
        - "The malicious Activity registers to receive another application's implicit Intents, and it is then started in place of the expected Activity"
        - False response activity
- Detection
    - "ComDroid issues a warning when it detects an implicit Intent being sent with weak or no permission requirements. Intents sent through the sink may be vulnerable to action- based attacks (e.g., broadcast denial of service or Activity/Service launching). If any of these Intents contain extra data, then they may also be vulnerable to eavesdropping."
- Prevention
    - "When sending private data, applications should use explicit Intents if possible. Internal communication can and should always use explicit Intents. If it is not possible to use explicit Intents, then the developer should specify strong permissions to protect the Intent. Results returned by other components in response to Intents need to be verified to ensure that they are valid results from an expected source."

Special intents – access to URI's (Unauthorized intent receipt)

- "Intents can include URIs that reference data stored in an application's Content Provider. In case the Intent recipient does not have the privilege to access the URI, the Intent sender can set the FLAG_GRANT_READ_URI_PERMISSION or FLAG_GRANT_WRITE_URI_PERMISSION flags on the Intent. If the Provider has allowed URI permissions to be granted (in the manifest), this will give the Intent recipient the ability to read or write the data at the URI. If a malicious component intercepts the

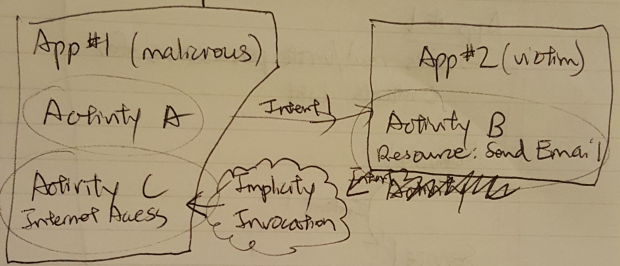
Intent (in the ways previously discussed), it can access the data URI contained in the Intent. “

- Detection using COVERT
  - “In short, the assertion states that the dst component (victim) has access to a permission (usesPermission) that is missing in the src component (malicious), and that permission is not being enforced in the source code of the victim component, nor by the application embodying the victim component. Recall from Section 5 that there are two ways of checking permissions in Android. The specified assertion relies on the specification of an intentResolver function, shown in Listing 9. The Component, Intent and IntentFilter signatures are specified such that they have all the necessary attributes required for Intent resolution. We thus describe intent-resolver as a function augmenting the aforementioned androidDeclaration module. This function takes as input an Intent and returns a set of Components that may handle the Intent under consideration. Given the Intent is explicit, it should be delivered to the recipient identified by the component field of the Intent (line 3). Otherwise, the resolver checks Components’ IntentFilters to find those whose elements are matched against the given Intent. Specifically, an implicit Intent must pass a matching test with respect to each of the action, data, and categories elements on the IntentFilters bound to a component (as stated in lines 6–9). Seeing that a component can define multiple IntentFilters, an Intent that does not match one of a component’s IntentFilters may match another (lines 4–5).”
- Prevetion
  - “If the data is intended to be private, then an Intent carrying data privileges should be explicitly addressed to prevent interception.”

Old assignment:

The example vulnerability is dependent on the user having/using a malicious app to begin with.

Another example



Basically any app that uses implicit intents could potentially be vulnerable in this way. The malicious app just needs to figure out what resource the implicit intent needs, and register itself as a malicious version of that resource. The victim can be invoked in any number of ways, such as broadcast (example), or like this.

Whatever info getting passed to the intent, e.g. pictures or emails or any other sensitive data could be hijacked, and the severity of the attack depends on what is leaked.

This can be stopped by simply not using malicious apps and avoiding implicit invocations

Users data can also be compromised if they give an app permissions that they shouldn't have given it.

App #1
↳ Gets read/write permissions for Contacts List

Activity 1
↳ Intent 1

Service 1
↳ Sends contacts data to unknown destination
↳ Deletes all contacts

This is an example of what could happen if an untrusted app gets permissions that it shouldn't. Any data that is entrusted with, i.e. contacts data here, could be compromised, made public, or altered / deleted.

This can be prevented by not giving unnecessary permissions to untrusted apps