Android Application Vetting

December 9, 2013

1 Comdroid

- Talks about Android components.
- Interactions through intents. Application processes run in separate protection domains. Intents are like IPC messages. Android framework takes care of unmarshalling arguments and invoking the call in target app.
- Threat Model: Applications are malicious. Especially the exported part. Application may export sensitive routines. Or may send intents to untrusted app that may steal information.
- Intent filters are described in manifest along with the component. When intents are sent, they can explicit mention component name, or mention the filter that an application has to satisfy. Filter categories are action, data category.

1.1 Malicious component is receiver

- Broadcast receivers can get broadcasts in an unauthroised way. If it can register as high priority receiver, it can receive intents and not propogate them resulting in DoS attack.
- Activity can steal data. Can spoof the interface and perform phisihing, false response attack, etc. Return malicious response value that changes program state.
- Service can steal data, phish, spoof. If application provides call backs can do additional damage.
- Special intents delegate intent permissions sent with a msg. They can be used to grant permission to write to Content Providers, or perform actions, which can be passed down a line. Can do damage to app's datastore.

1.2 Malicious component is sender: Intent spoofing

- Broadcast receivers can be tricked to perform senstive tasks by a spoofed intent.
- Malicious application can change program state, when sending an intent to an activity.

  - Malicious application can change program state, when sending an intent to a service. Can use service's methods carelessly exposed to do wicked stuff. Service can return sensitive information back to the caller.

1.3 Verification

- Track intent objects, filters, data, permissions attached and track if sent inappropriately as an implicit intent at various sinks.
- Track components in manifests and see if they are not unwittingly exported.

2 CHEX

- Since exported components, are a vital part of android, it is important that they are safe. A component might be unwittingly exported. Declaring intent filters forces a component to be exported.
- CHEX looks at exported components, identifies different event handlers that can be called by android framework. It tracks data flows, dividing the control flow from entry points to splits.
- The data flow from sensitive sources to sinks can occur during an asynchronous call the component serves. For example during one call, a service may get Location information, store it in a instance variable, and then for a next call send it over a network. CHEX has to find when more than one flows contribute to leak.
- The flow of information from sensitive sources, to sensitive sinks is tracked. In a class, it can flow through instance variables or pair funcions (setters followed by getters).
- CHEX permutes splits and performs flow analysis, to see if any execution can cause sensitive information to be read from an app, written to an app or both.