Securing Embedded User Interfaces: Android and Beyond

Franziska Roesner and Tadayoshi Kohno

University of Washington
Embedded User Interfaces

Embedded third-party UIs are common on websites and in smartphone apps.

On the Web: via iframes

```html
<iframe src="https://maps.google.com/..."></iframe>
```
Embedded User Interfaces

Embedded third-party UIs are common on websites and in smartphone apps.

On Android: include library code
Security and Embedding

Browsers provide secure isolation between an embedding page and embedded content.

Android does not.

• Third-party libraries run in app’s context.
• No true cross-application UI embedding.
Outline

• The Case for Secure UI in Android
• Design & Implementation: LayerCake
• Evaluation
  – Functionality case studies
  – Performance
• Summary
Outline

• The Case for Secure UI in Android
• Design & Implementation: LayerCake
• Evaluation
  – Functionality case studies
  – Performance
• Summary
Security Concerns on Android

Both the parent and the child may be malicious.
Security Concerns: Malicious Child

Example: Screen takeover (or redirection)

Ad Library Code

```java
View parent = adView.getParent();
parent.removeChildChildren();
parent.addChild(fullScreenAd);
```

Code in the same context can access all UI elements.
Security Concerns: Malicious Parent

Example: Input Eavesdropping and Blocking

Input events propagate down the UI layout tree, through potentially untrusted nodes.
Many Security Concerns

Malicious parents and children can both perform:
- Data theft, Display forgery, Focus stealing,
- Programmatic input forgery

Additionally, a malicious parent can perform:
- Input eavesdropping, Input DoS,
- Size manipulation, Clickjacking

Additionally, a malicious child can perform:
- Ancestor redirection
This Work

Many (though not all) of these attacks are impossible with iframes on the Web.

Most of these attacks are possible on Android.

• Existing approaches [AdDroid: Pearce et al., AdSplit: Shekhar et al.] only target ad scenario.
• Our prior work [UIST '12] considered secure UI embedding in theory.

What does it take to implement secure third-party embedding on Android?
Outline

• The Case for Secure UI in Android

• Design & Implementation: LayerCake

• Evaluation
  – Functionality case studies
  – Performance

• Summary
Secure UI Embedding for Android

LayerCake is a modified version of Android 4.2 (Jelly Bean) that securely supports embedded applications.
Android Background

• **Activity**: A page of an application’s UI.
  – Only one Activity in the foreground at a time.
  – Activity consists of tree of UI elements (**Views**).

• Activity drawn in a **Window**.
  – Contains one View tree.
Supporting Embedded Activities

**Goal:** Allow an Activity in one application to securely embed an Activity from another app.

1. Separate processes.
2. Separate windows.
3. Handle additional security concerns.

Requires pervasive changes to ActivityManager and WindowManager.
(1) Separate Processes

Allow developers to embed Activities from other applications ("iframes for Android").

Challenges:

- Separating code into processes prevents direct UI manipulation.
- Multiple running Activities
- Parent-child communication
Separate Processes Not Sufficient

How does LayerCake actually embed cross-application UI?

Diagram showing the connection between WindowManager, Relative Layout, Embedded ActivityView, Frame Layout, and user input.
(2) Separate Windows

Visually overlay separate windows, don’t nest UI trees.

Separating UI trees prevents input eavesdropping and DoS attacks.

Visually overlay child window on parent window.
Overlaying: Practical Challenges

Layout changes must be automatically propagated across processes. Cropping is needed to make overlaying look like embedding.
(3) Additional Security: Handling Size Conflicts

**Threat:** What if the parent makes the child too small? (e.g., camera preview)

**Observation:** Enforcing a minimum size provides no additional security on its own: attacker can mimic effect by scrolling or obstructing.
(3) Additional Security: Preventing Clickjacking

**Threat:** Trick user into clicking on an embedded element that is **visually obscured**.

Embedded Activities can request to **NOT receive user input events** if they are:

1. Covered (fully or partly) by another window.
2. Not the minimum requested size.
3. Not fully visible due to window placement.

(Additional clickjacking protection: e.g., *InContext: Huang et al.*)
(3) Additional Security: Preventing Ancestor Redirection

**Threat:** What if a malicious child tries to open a new top-level Activity?

- **Note:** Opening another *embedded* Activity (in its place) is ok.

- **On attempt to open top-level Activity:**
  - Prompt user, or
  - Allow automatically if in response to user click ($\approx$ user intent)
Outline

• The Case for Secure UI in Android
• Design & Implementation: LayerCake
• Evaluation
  – Functionality case studies
  – Performance
• Summary
Functionality Case Studies

Not (securely) possible on stock Android; enabled by LayerCake:

Advertising

Facebook Widgets

User-Driven Access Control [Oakland ’12]

Apply to top-level redirection.
Legacy Applications

Applications don’t require modification to be embedded.
Performance Evaluation: Activity Load Time

<table>
<thead>
<tr>
<th>Application</th>
<th>Load time (10 trial average)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Embedding</td>
</tr>
<tr>
<td>RestaurantReviewer</td>
<td>163 ms</td>
</tr>
<tr>
<td>FacebookDemo</td>
<td>158 ms</td>
</tr>
<tr>
<td>Listen&amp;Shop</td>
<td>160 ms</td>
</tr>
</tbody>
</table>

* Note that load time for parent Activity is unaffected.
Performance Evaluation: Event Dispatch

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Event Dispatch Time (10 trial average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock Android</td>
<td>1.9 ms</td>
</tr>
<tr>
<td>No focus change</td>
<td>2.1 ms</td>
</tr>
<tr>
<td>Focus change</td>
<td>3.6 ms</td>
</tr>
</tbody>
</table>
Outline

• The Case for Secure UI in Android
• Design & Implementation: LayerCake
• Evaluation
  – Functionality case studies
  – Performance
• Summary
Contributions

**LayerCake**: Artifact resulting from systematic application of secure embedded UI concepts.


Lessons Learned:

- Visually overlay windows, don’t nest UI trees.
- Size manipulation, scroll placement, and obstruction must be considered together.
- Ancestor redirection can follow user intent.
Summary

• Embedded third-party UIs pose security concerns, unaddressed on Android.

• LayerCake: modified version of Android that securely supports application embedding.

• See me for demo!

http://layercake.cs.washington.edu