A Tale of Two Cities: How WebView Induces Bugs to Android Applications

Jiajun Hu, Lili Wei, Yepang Liu, Shing-Chi Cheung, and Huaxun Huang
WebView – What is it?

WebView is an Android component that augments a native app with web browser capabilities.
WebView – Popularity

It is widely used in real-world Android applications - 998286/1172610 (85%) Google Play apps use WebView in some fashion [Mutchler et al. MoST’15]
WebView – Inducing bugs!

WebView-induced bugs are common in Android apps

Comments from real-world Android apps:

“The WebView in that view gave me enough headaches already”

“WebView might have caused crashes on some devices”

“WebView fails to show images in private post”
Existing Work

Existing work focuses mainly on the security issues induced by WebView

Goal: understand how WebView induces bugs to real-world Android applications
Overview

• **Empirical Study**: understanding common features of WebView-induced bugs (ωBugs for short)

• **Tool (ωDroid)**: detecting common types of ωBugs

• **Evaluation**: evaluating the bug detection ability of ωDroid
Overview

• **Empirical Study**: understanding common features of WebView-induced bugs (ωBugs for short)

• **Tool (ωDroid)**: detecting common types of ωBugs

• **Evaluation**: evaluating the bug detection ability of ωDroid
Empirical Study – Research Questions

• RQ1: Bug Cause
  • What are the common root causes of ωBugs?

• RQ2: Bug Consequence
  • What are the common consequences of ωBugs?

• RQ3: Bug Manifestation
  • How do ωBugs manifest themselves? Can we propose testing technique to effectively expose ωBugs?
Empirical Study – Research Questions

• **RQ1: Bug Cause**
  • What are the common root causes of ωBugs?

• **RQ2: Bug Consequence**
  • What are the common consequences of ωBug?

• **RQ3: Bug Manifestation**
  • How do ωBugs manifest themselves? Can we propose testing technique to effectively expose ωBugs?
Empirical Study – Methodology

F-Droid

Include WebView

293 apps

Bug reports

Manual inspection

124 ωBugs

From 51 apps

Existing papers

Contain keywords: webview & javascript
Empirical Study – RQ1: Bug Cause

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misaligned WebView Lifecycles✓</td>
<td>35</td>
</tr>
<tr>
<td>WebView Evolution &amp; Device Customization✓</td>
<td>40</td>
</tr>
<tr>
<td>Misconfiguration✓</td>
<td>21</td>
</tr>
<tr>
<td>API Misuse</td>
<td>16</td>
</tr>
<tr>
<td>Bridge Communication</td>
<td>4</td>
</tr>
<tr>
<td>WebView Limitation</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>124</td>
</tr>
</tbody>
</table>

ωBugs root cause distribution
Empirical Study – RQ1: Bug Cause

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misaligned WebView Lifecycles</td>
<td>35</td>
</tr>
<tr>
<td>WebView Evolution &amp; Device Customization</td>
<td>40</td>
</tr>
<tr>
<td>Misconfiguration</td>
<td>21</td>
</tr>
<tr>
<td>API Misuse</td>
<td>16</td>
</tr>
<tr>
<td>Bridge Communication</td>
<td>4</td>
</tr>
<tr>
<td>WebView Limitation</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>124</td>
</tr>
</tbody>
</table>

ωBugs root cause distribution

Managed by the Android OS

Explicitly managed by app developers

Activity lifecycle & WebView lifecycle
Empirical Study – RQ1: Bug Cause

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misaligned WebView Lifecycles</td>
<td>35</td>
</tr>
<tr>
<td>WebView Evolution &amp; Device Customization</td>
<td>40</td>
</tr>
<tr>
<td>Misconfiguration</td>
<td>21</td>
</tr>
<tr>
<td>API Misuse</td>
<td>16</td>
</tr>
<tr>
<td>Bridge Communication</td>
<td>4</td>
</tr>
<tr>
<td>WebView Limitation</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>124</td>
</tr>
</tbody>
</table>

Root Cause Distribution

- WordPress issue 484:

```java
protected void onPause()
{
    super.onPause();
    mWebView.onPause();
}
```

Video

ASMR videos. For those of you who don’t know what ASMR (Autonomous Sensory Meridian Response) is, it’s an experience characterised by a tingling sensation that typically begins on the scalp and it’s triggered by sound and images.

I’ve been loving this kind of videos for a long time, specially to relax after a long day. So I’ve finally been brave enough to create my own channel called Glam ASMR. If you haven’t seen any video of this kind, it can look a bit weird, because I’m whispering in order to relax. Some people don’t even get the tingling sensation, but...
Empirical Study – RQ1: Bug Cause

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misaligned WebView Lifecycles</td>
<td>35</td>
</tr>
<tr>
<td>WebView Evolution &amp; Device Customization</td>
<td>40</td>
</tr>
<tr>
<td>Misconfiguration</td>
<td>21</td>
</tr>
<tr>
<td>API Misuse</td>
<td>16</td>
</tr>
<tr>
<td>Bridge Communication</td>
<td>4</td>
</tr>
<tr>
<td>WebView Limitation</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>124</td>
</tr>
</tbody>
</table>

- ωBugs can cause severe compatibility problems
  - WebView is fast evolving
  - It is non-uniformly supported on different device models

ωBugs root cause distribution
# Empirical Study – RQ1: Bug Cause

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misaligned WebView Lifecycles</td>
<td>35</td>
</tr>
<tr>
<td>WebView Evolution &amp; Device Customization</td>
<td>40</td>
</tr>
<tr>
<td>Misconfiguration</td>
<td>21</td>
</tr>
<tr>
<td>API Misuse</td>
<td>16</td>
</tr>
<tr>
<td>Bridge Communication</td>
<td>4</td>
</tr>
<tr>
<td>WebView Limitation</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>124</strong></td>
</tr>
</tbody>
</table>

- Many web pages require customized WebView settings to function properly

```javascript
WebSettings.
setJavaScriptEnabled(true)
```

- Non-trivial to correctly configure WebView

Over 50 settings

Don’t know the page to be loaded in advance
Empirical Study – RQ3: Bug Manifestation

• Manifestation of ωBugs can be content-sensitive (24%)

- Image
- Animation
- Video/audio
- Character encodings

WordPress issue 484
Empirical Study – RQ3: Bug Manifestation

• Specific sequences of lifecycle events are required to trigger common ωBugs (22%)
Empirical Study – RQ3: Bug Manifestation

- Effective oracles can be proposed for automated detection of ωBugs
- Comparing WebView UI displays and resource usages before and after triggering lifecycle events (20%).

UI Inconsistency

Unpaused Resources
Empirical Study – RQ3: Bug Manifestation

• Effective oracles can be proposed for automated detection of ωBugs
  • Comparing the behaviors of WebView and web browser (10%)
Overview

• Empirical Study: understanding common features of ωBugs

• Tool (ωDroid): detecting common types of ωBugs

• Evaluation: evaluating the bug detection ability of ωDroid
ωDroid – ωBugs Testing

- The design of ωDroid is based on 2 major observations.
  - Obs 1: Lifecycle events are needed to trigger ωBugs. Bug-triggering operation
  - Obs 2: ωBugs can be identified by comparing UI displays and resource usages before and after lifecycle events happen. Oracle

![Diagram showing the test event generator and oracle process.]

1. Test Event Generator (on top of Monkey)
2. Lifecycle Event Injector
3. Lifecycle Misalignment Oracle
4. Report Bug

Obs 1

App

Obs 2
ωDroid – Lifecycle Event Injector

Injecting 3 sequences of lifecycle events:

• Pause – resume
• Stop – restart
• Destroy – recreate
ωDroid – Lifecycle Misalignment Oracle

Activity restart → UI Inconsistency

Activity pause → Unpaused Resources

Oracle 2018
The displayed content of a WebView after activity restart should be **consistent** with that before restart.
The displayed content of a WebView after activity restart should be consistent with that before restart.

Resources loaded by a WebView should be paused when its enclosing activity is paused.
ωDroid – Lifecycle Misalignment Oracle

- UI Inconsistency

Node matching

UI Model (DOM tree) before activity restart

UI Model (DOM tree) after activity restart
ωDroid – Lifecycle Misalignment Oracle

• Unpaused Resources

Detect unpaused **video** resource by testing sound from video
Overview

• Empirical Study: understanding common features of ωBugs

• ωDroid: detecting common types of ωBugs

• Evaluation: evaluating the bug detection ability of ωDroid
Evaluation – Subjects & Setup

Subjects = 293 apps
- 51 apps formed the empirical study dataset
- Apps that have no revision within the past 6 months = 146 apps

- Running time: 1 hour
- Framework: Android emulator running Android 6.0 (Marshmallow)
- 2 baselines:
  - Baseline 1: Test Event Generator
  - Baseline 2: Test Event Generator + Lifecycle Event Injector

\[ \text{Crash} \]
Evaluation – Effectiveness and Practicality

• ωDroid reported 36 distinct ωBugs in 31 apps
• 30 TPs (83.3%)

We reported all the TPs to the corresponding app developers.

• 14 bugs have been confirmed, 7 of them have been quickly fixed.
Evaluation – Usefulness of Lifecycle Events & Oracles

• ωDroid

• Baseline 1: Test Event Generator

• Baseline 2: Test Event Generator + Lifecycle Event Injector
Evaluation – Usefulness of Lifecycle Events & Oracles

- oDroid
  - 30 true bugs
- Baseline 1: Test Event Generator
  - 0 WebView-related crash
- Baseline 2: Test Event Generator + Lifecycle Event Injector
  - 1 WebView-related crash

The proposed oracles are useful in identifying ωBugs.
Without effective oracles, injecting lifecycle events alone has limited effect on exposing ωBugs
In future:

- Test input generation: appropriate web contents
- Differential testing: comparing with web browser
Our dataset is publicly available online

http://home.cse.ust.hk/~jhuao/wDroid.html