Seamless In-App Ad Blocking on Stock Android

Michael Backes, Sven Bugiel, Philipp von Styp-Rekowsky, and Marvin Wißfeld
CISPA, Saarland University

Mobile Security Technologies (MoST) Workshop
San Jose, California, May 25, 2017
Motivation

Ads allow developers to easily monetize their apps.
Why to block ads on Android?

Ad libraries have shown to exploit host app’s permissions to access private data.

Ads can be used to lure users into installing malware.

Streaming media files can be expensive on mobile networks.
Existing approaches lack deployability or effectiveness.
## Existing approaches

<table>
<thead>
<tr>
<th>Functional Objectives</th>
<th>OS extension(^1)</th>
<th>App rewriting(^2)</th>
<th>Network filter(^3)</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1: No system modification</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>O2: No application modification</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>O3: Blocking cached/pre-packaged ads</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
</tr>
</tbody>
</table>

\(\checkmark\) = applies; \(\times\) = does not apply.

\(^1\) **AdDroid** Pearce et al, ASIACCS’12, **Adsplit** Shekhar et al, Usenix’12, **Aframe** Zhang et al, ACSAC’13

\(^2\) **PEDAL** Liu et al, MobiSys’15, **Apklancet** Yang et al, ASIACCS’14

\(^3\) **Privacyguard** Song et al, SPSM’15, **Adguard**
Contribution

We developed an in-app ad blocking system, that

- is easy to deploy and runs on-device only.
- effectively blocks ad library code execution.
- has no side-effects on the applications.
Ad library inclusion
Approach

1. Identification
   Find ad library API classes inside app package

2. Stub generation
   Create matching classes that preserve functionality

3. Injection
   Have the application use the created stub
Approach: 1. Identification

Find ad library API classes inside app package

Task: identify class AdView from included library com.example.ads.

Approach: find class with class name com.example.ads.AdView.
Problem: Identifier Renaming

Build process obfuscates names of classes, methods and fields:

com.example.ads.AdView -> a.b.a.a
com.example.adsInterstitialAd -> a.b.a.b

... but when referenced in XML or string constants, names are preserved.
- Libraries contain a XML manifest referencing their package name.
- UI classes might be referenced in UI XML.

com.example.ads.AdView -> com.example.ads.AdView
com.example.adsInterstitialAd -> com.example.ads.a
Task: identify class AdView from included library com.example.ads.

Approach: find class in package com.example.ads with same superclass and members.
Problem: Dead code elimination

Build process removes methods and classes that are not referenced.
Task: identify class AdView from included library com.example.ads.

Approach: find class in package com.example.ads with same superclass and required members.
Filter rules

Must contain for each class: package name, superclass, required members.

```java
package com.example.ads
class .AdView extends android.view.View
    method exists void loadAd
end class
class .InterstitialAd
    method exists void openAd .AdListener
end class
class .AdListener
    method exists void onAdClosed
end class
end package
```
Approach: 2. Stub generation

Create matching classes that preserve functionality

Task: create class replacing `InterstitialAd`.

Approach: Replace all methods with empty/null-return methods.
Problem: Callbacks

Some method calls must result in callback invocations to preserve app functionality
Task: create classes replacing `InterstitialAd`.

Approach: Replace all methods with empty/null-return methods or functionality preserving implementations.
Filter rules

Must contain for each class: package name, superclass, required members, stub generation info.

```java
package com.example.ads
class AdView extends android.view.View
    set filter-action empty-view
    method exists void loadAd
end class
class InterstitialAd
    set filter-action empty-object
    method exists,replace void openAd .AdListener
end class
class AdListener
    method exists void onAdClosed
end class
end package
```
Approach: 3. Injection

Have the application use the created stub

Use app virtualization (Boxify Backes et al., Usenix’15) to instrument app.

Prepend stub classes to class loader search path, so they are loaded first.
Evaluation

Manual assessment

Created filter rules for 7 large advertisers

Tested against 22 random apps from Play Store (that contained ads)

Ads blocked in 19 apps, 3 failed because of missing filter rules.

No app crashed or misbehaved.
Real-world test

Made end-user version (with more filter rules) publicly available

5,700+ installs, 15,000+ different apps ad-blocked

Less than 200 reported apps that still showed ads.
Limitations

- Only third-party libraries. This excludes
  - Content ads (ex. Spotify, Facebook)
  - Web-based ads (WebView, Browser)
- Dynamic class loading
- Stronger obfuscation
- Red Pill attacks
Summary

Built in-app ad blocking based on app virtualization.
Demonstrated deployability and efficiency by real-world evaluation.

www.srt-adversary.com