Iframes/Popups Are Dangerous in Mobile WebView: Studying and Mitigating Differential Context Vulnerabilities

Guangliang Yang, Jeff Huang, and Guofei Gu

*Secure Communication and Computer Systems Lab
Texas A&M University
Iframes/Popups in Regular Browsers

- In modern web apps, iframes/popups are frequently used. Their security has been well studied in regular browsers.
Iframes/Popups in Regular Browsers

- In modern web apps, iframes/popups are frequently used. Their security has been well studied in regular browsers.

- However, the security study on a new web environment, called mobile WebView, is still missing.
WebView

- An embedded browser-like UI component in mobile apps (i.e., hybrid apps)
- Easy to use and powerful
- Frequently used by mobile apps
  - Integrated in ~80% Android apps
Motivation & Our Work

- WebView provides a totally new working environment for iframes/popups.

=> Are iframes/popups still safe in WebView?
Motivation & Our Work

- We conduct the first security study in Android WebView => *Differential Context Vulnerabilities* (DCVs)

- We assess the security impacts on real-world apps with DCV-Hunter:
  - Facebook, Instagram, Facebook Messenger, Google News, Skype, Uber, Yelp, and U.S. Bank …

- We propose a novel multi-layer defense solution.
Security Study & DCV
Threat Model

- Mobile code is benign
- WebView may contain untrusted content
  - The main (top) frame is trusted
  - Iframes/popups loading third-party content are untrusted.
Possible Attacks: Untrusted iframes/popups may trigger and leverage these inconsistencies to obtain risky abilities.
Inconsistencies Between Browsers and WebView

- UI Design Styles

(a) Regular Browser UI

(b) WebView UI
Security Issues & Concrete Attacks

- The lack of the address bar

  => **Main-Frame Navigation Attacks**: Untrusted iframes/popups launch phishing attacks by secretly navigating the main frame.

- Permissive navigation policy
  - Any sub-frame can navigate the main frame
  - Not harmful in regular browsers
    - iframe sandbox + address bar
  - But dangerous in WebView
Security Issues & Concrete Attacks

- Example: A banking app
Security Issues & Concrete Attacks

- Example: A banking app

![Reset Password](image)
Security Issues & Concrete Attacks

- Example: A banking app

```javascript
window.open("http://attacker.com", "_top")
```
Security Issues & Concrete Attacks

- Example: A banking app

![Malicious webpage](image-url)
Security Issues & Concrete Attacks

- The lack of the tab bar

- Principles
  - Each web window is rendered by an independent WebView UI

=> **WebView UI (WUI) Redressing Attacks:**
Untrusted iframes/popups launch phishing attacks by creating a malicious WUI and overlapping begin WUI with the new WUI.
Security Issues & Concrete Attacks

- WUI Redressing Attacks
Security Issues & Concrete Attacks

- WUI Redressing Attacks
  - Possible Attack #1: Overlap attack
    - Manipulating the rendering order of multiple WUIs

![Diagram showing WUI #2 with untrusted web content and a pop-up window leading to WUI #1 with a benign web page and an untrusted sub-frame.](image-url)
Security Issues & Concrete Attacks

- WUI Redressing Attacks
  - Possible Attack #2: Closure attack

![Diagram of WUI Redressing Attacks]

- Untrusted Web Content
- Benign Web Page
- Untrusted Sub-Frame
Security Issues & Concrete Attacks

- WUI Redressing Attacks
  - Possible Attack #2: Closure attack

```javascript
window.close
```
Security Issues & Concrete Attacks

- Example: a flight searching app
Security Issues & Concrete Attacks

- Example: a flight searching app
Security Issues & Concrete Attacks

- Example: a flight searching app

![Flight ticket example](image)
Security Issues & Concrete Attacks

- Example: a flight searching app

malicious webpage

You chose a Basic Economy fare

Basic Economy
Most restrictions

- Seats assigned at check-in
- Not eligible for upgrades
- No flight changes or refunds
- Board in last group

*Rules apply to all passengers including AAdvantage elites.

Accept restrictions
Inconsistencies Between Browsers and WebView

- Programming features
  - WebView enables many programming APIs to let developers customize their own WebView instances.

  `WebView.setSupportMultipleWindows(true/false)`
Security Issues & Concrete Attacks

- WebView customization vs. Regular web behaviors

=> Privileged main-frame navigation attack

- WebView.SupportMultipleWindows = false

- `window.open("https://attacker.com", "_blank")`

- Iframe sandbox?  **No!**
DCV Summary

- WebView UI Redressing Attacks
  - Creation & Closure
- Main-Frame Navigation Attacks
  - Traditional & Privileged

- Origin Hiding Attacks

- Existing defense solutions are limited to prevent DCV based attacks.
Security Assessment
DCV-Hunter: Automatic Vulnerability Detection
Security Assessment

- Dataset
  - 17K most popular free apps from Google Play
    - = 32 categories X 540 apps for each category

- Result overview
  - 11,341 hybrid apps
  - 4,358 hybrid apps (38.4%) were potentially vulnerable, including
    - 13,384 potentially vulnerable WebView instances and
    - 27,754 potential vulnerabilities
  - 19.5 Billion downloads

- Low false positive
Security Assessment

<table>
<thead>
<tr>
<th>Potential Attacks</th>
<th>Impacted WebView</th>
<th>Impacted Apps</th>
<th>App Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin-Hiding</td>
<td>1,737</td>
<td>1,238</td>
<td>3.5 Billion</td>
</tr>
<tr>
<td>WUI Overlap</td>
<td>138</td>
<td>89</td>
<td>8 Billion</td>
</tr>
<tr>
<td>WUI Closure</td>
<td>5</td>
<td>5</td>
<td>13 Million</td>
</tr>
<tr>
<td>Traditional Navigation</td>
<td>13,384</td>
<td>4,358</td>
<td>19.5 Billion</td>
</tr>
<tr>
<td>Privileged Navigation</td>
<td>12,490</td>
<td>4,161</td>
<td>17.8 Billion</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>13,384</strong></td>
<td><strong>4,358</strong></td>
<td><strong>19.5 Billion</strong></td>
</tr>
</tbody>
</table>
Security Assessment

- Many high-profile apps are impacted
  - Facebook, Instagram, Facebook Messenger, Google News, Skype, Uber, Yelp, WeChat, Kayak, ESPN, McDonald’s, Kakao Talk, and Samsung Mobile Print

- Third-party development libraries
  - Facebook Mobile Browser & Facebook React Native

- Leading password management apps
  - dashlane, lastpass, and 1password

- Popular banking apps
  - U.S. bank, Huntington bank, and Chime mobile bank
Case Studies

- Facebook Messenger
  - Providing its own address bar?
  - No! pixel & race-condition problems
Case Studies

- Facebook Messenger
Case Studies

- Facebook Messenger
- WUI redressing attack
Case Studies

- Facebook Messenger
  - WUI redressing attack
Case Studies

- Facebook Messenger
  - *Blended attack: WUI redressing attack + Traditional navigation attack*
Case Studies

- Facebook Messenger
  - *Blended attack: WUI redressing attack + Traditional navigation attack*

![Ebay.com]

- Pop-up
  - 2) Refresh the old WUI!
Case Studies

- Facebook Messenger
  - *Blended attack: WUI redressing attack + Traditional navigation attack*

Demos: https://sites.google.com/view/dcv-attacks
DCV Mitigation
DCV Mitigation

- Mitigating the DCV issues from the root (i.e., inconsistencies)
  - Reducing the inconsistencies between browsers and WebView
    - Floating URL address bar
    - Validating sensitive operations (e.g., popup creation)

- Evaluation
  - Our defense solution is
    - Effective
    - Compatible (90% Android devices)
    - Low-overhead
Conclusion
Conclusion

- WebView attracted more and more attention.

- Iframe/popup behaviors were well studied in regular browsers, but rarely understood in the new web environment of WebView.

- We filled the gap by identifying a novel class of vulnerabilities (DCVs), assessing the security impacts with a novel detection tool (DCV-Hunter), and mitigating the DCV issues with a multi-layer defense solution.
Thanks!
Inconsistencies Between Browsers and WebView

- Programming features
Inconsistencies Between Browsers and WebView

- Programming features

Diagram:
- WebView
  - Main Frame
  - Sub-Frame
  - Web-Mobile Bridge
  - Sensitive Functionalities
  - WebView Setting APIs
  - WebView Content Loading APIs
  - Native (e.g., Java)

- Benign Hybrid App

WebView.Support-MultipleWindow()
Inconsistencies Between Browsers and WebView

- Programming features

```
WebView.loadUrl()
```