Identity Confusion in WebView-based Mobile App-in-app Ecosystems

Lei Zhang\textsuperscript{1,*}, Zhibo Zhang\textsuperscript{1,*}, Ancong Liu\textsuperscript{1}, Yinzhi Cao\textsuperscript{2}, Xiaohan Zhang\textsuperscript{1}, Yanjun Chen\textsuperscript{1}, Yuan Zhang\textsuperscript{1}, Guangliang Yang\textsuperscript{1}, Min Yang\textsuperscript{1}

1. Fudan University
2. Johns Hopkins University

*: The first two authors have contributed equally to this work.
App-in-App Ecosystem

- **Super-app**
  - A mobile app with rich functionalities, often delegate their functions to other parties (sub-apps)
  - e.g., Paytm, Snapchat, TikTok, WeChat

- **Sub-app**
  - brings rich content and services
  - native app like experience

source: https://paytm.com
The Popular Trend

• More than 47 high-profile super-apps
  • world-wide
    • Asia (Grab, Line, Paytm), European (VK), and the U.S. (Microsoft Teams)...
  • large user base
    • with 46B+ downloads in total

• Huge amount of sub-apps
  • e.g., 3.8M+ sub-apps in WeChat

source: https://paytm.com
Programming Model & Lifecycle

• Program language
  • JavaScript, HTML, and CSS
• Upload to super-app market
  • driven by URI
    • e.g., super-app://sub-appID/path/query
• Load multi-party resource
  • from sub-app market
  • from third-party servers
• Access privileged APIs

Crucial Question: determining who can call specific privileged APIs
A Survey Study

• Popular Super-app Runtimes
  • an embedded browser instance
    • (customized) WebView in Android
    • WKWebView in iOS
  • web-to-mobile bridge
    • enable JS to call Java functions
• runtime APIs
  • 50% are un-documentated
  • 80% are privileged
    • access user data
      • e.g., account, bank info, phone number…
    • access OS resources
      • e.g., camera, location…

---

[Diagram of system services and web-to-mobile bridge]
A Survey Study

• Existing Identity Checks
  • Domain Name
    • represents a server and contents delivered from the server
  • App ID
    • assigned by a super-app to the sub-app
  • Capability
    • a secret issued by either a super-app or a server and checked based on exact match.
Identity Confusion

• Definition
  • intended identity can be broader than or different from it actually represents
  • disobey the least privilege principle

• Domain Name Confusion
  • Privileged web domain in an unprivileged sub-app

• App ID Confusion
  • Unprivileged web domains in a privileged sub-app

• Capability Confusion
  • Privileged capability obtained by an unprivileged sub-app or domain
Identity Confusion Attack

• Once exploited, the attacker can
  • inject phishing web page to popular sub-app
  • steal user data (address, payment info, phone number, email...)
  • abuse OS resources (open microphone, install malicious apk)

I am an attacker!

phishing deeplink

steal email, phone number

Malicious Web Content

1688 sub-app

steal address, payment info
A Taxonomy Study

• Domain Name Confusion
  • happens in domain based identity check
  • checked domain != actual domain

Expected Identity Check

Flawed Identity Check

check malicious.com

malicious.com -> privileged APIs

check privileged.com

malicious.com -> privileged APIs
Domain Name Confusion

• **Type 1: Timing-based Confusion**
  • Case 1. super-apps use `onPageStarted()` to get identity
    • race condition between different threads of WebView

Figure: Race between WebView’s Render and Browser Threads
Domain Name Confusion

• **Type 1: Timing-based Confusion**
  • Case2. super-apps use `getUrl()` to get identity
    • race condition between different threads of super-app

![Diagram](image)

**Figure: Race between super-app’s Dispatch and Check Threads**
## Domain Name Confusion

- **Type 2: Frame-based Confusion**
  - an iframe acts on behalf of the top frame’s identity

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Method Signature of Event Handlers</th>
<th>Domain Name Confusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Timing-based</td>
</tr>
<tr>
<td><strong>Getter Method:</strong></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>WebView</td>
<td>getOriginalUrl ()</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>getUrl ()</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Callback Method:</strong></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>WebViewClient</td>
<td>doUpdateVisitedHistory (WebView view, String url, boolean isReload)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onLoadResource (WebView view, String url)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onPageCommitVisible (WebView view, String url)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onPageFinished (WebView view, String url)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onPageStarted (WebView view, String url, Bitmap favicon)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onReceivedClientCertRequest (WebView view, ClientCertRequest request)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onReceivedError (WebView view, WebResourceRequest request, WebResourceError error)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onReceivedHttpAuthRequest (WebView view, HttpAuthHandler handler, String host, String realm)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onReceivedHttpError (WebView view, WebResourceRequest request, WebResourceResponse errorResponse)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>shouldInterceptRequest (WebView view, WebResourceRequest request)</td>
<td>✓</td>
</tr>
<tr>
<td>WebChromeClient</td>
<td>shouldOverrideUrlLoading (WebView view, WebResourceRequest request)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>onReceivedTouchIconUrl (WebView view, String url, boolean precomposed)</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 3: The domain name confusion in using WebView’s event handlers to obtain identity information. We measure them at time and frame dimensions.
Prevalence & Consequence

- **Step II: Vulnerability Analysis**
  - **Methodology**
    - static analysis on super-apps to find whether a vulnerable API is used
    - write test cases and exploits
  - **Cross Platform Verification**
    - use the Proof of Vulnerability (PoV) for Android versions of super-apps to verify their iOS versions

```javascript
1  //JavaScript
2  window.setInterval(function(){
3      res = nativeInterface.framelessPostMessage('{}
4          "id":1,"func":"authentication.getAuthToken","
5          args":{"privileged.com"}]}});
6  //res can be leaked to malicious server
7  ... ...
8  },1500);
9  window.location.href = "https://privileged.com/#";
```

Figure: Example for verifying domain name confusion.
Prevalence & Consequence

• Step II: Vulnerability Analysis
  • Confusion Overall Result
    • *all* vulnerable to at least one type of identity confusion attack

Table 5: Breakdown of Identity Confusion Vulnerabilities of 47 Super-apps

<table>
<thead>
<tr>
<th>Identity Confusion</th>
<th># Super-apps</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type 1: Timing-based</td>
<td>15</td>
<td>WeChat, Alipay</td>
</tr>
<tr>
<td>Type 2: Frame-based</td>
<td>15</td>
<td>Microsoft Teams, Go-Jek</td>
</tr>
<tr>
<td>Total</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>AppID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type 1: Flawed matching</td>
<td>26</td>
<td>TikTok, Baidu</td>
</tr>
<tr>
<td>Type 2: Flawed parsing</td>
<td>2</td>
<td>WeChat, Go-Jek</td>
</tr>
<tr>
<td>Type 3: Missing checks</td>
<td>10</td>
<td>Microsoft Teams, UnionPay</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>Capability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type 1: Client-side</td>
<td>1</td>
<td>UnionPay</td>
</tr>
<tr>
<td>Type 2: Server-side</td>
<td>1</td>
<td>WeChat</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>No identity checks</td>
<td>9</td>
<td>Snapchat, Kuaishou</td>
</tr>
<tr>
<td>Total</td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>
Prevalence & Consequence

• **Step III: Consequence Analysis**
  • privilege escalation
  • phishing
  • privacy leaks

Table 8: Breakdown of Identity Confusion Consequences of 47 Super-apps

<table>
<thead>
<tr>
<th>Consequences</th>
<th># Super-apps</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privilege Escalation</td>
<td>38</td>
<td>Go-Jek, Grab</td>
</tr>
<tr>
<td>Phishing</td>
<td>31</td>
<td>TikTok, WeChat</td>
</tr>
<tr>
<td>Privacy Leaks</td>
<td>35</td>
<td>Alipay, Microsoft Teams</td>
</tr>
</tbody>
</table>
More in the Paper

• Two other identity confusions
  • App ID Confusion
  • Capability Confusion

• Overall Result Details
  • flaws & consequences in total 47 super-apps
  • other three consequences
    • permission re-delegation
    • data leakage
    • data over-collection

• Real-world Case Studies

• Mitigation & Discussion
Conclusion

• Conduct the first systematic study on identity confusion vulnerabilities in WebView-based app-in-app ecosystem.

• We collect and analyze 47 popular real-world super-apps, and confirms that they are all vulnerable to different types of identity confusion vulnerabilities.

• We thoroughly study why such identity confusion vulnerabilities exist and propose corresponding mitigation strategies based on the causes.
Thanks!

Q&A

zhibozhang19@fudan.edu.cn