What’s the Mini-App paradigm?

**Host App:**
- Act like an OS
- Fully manage mini-apps
- Provide system resources to mini-apps

**Mini-App:**
- Run inside the host app
- Call mini-app APIs
- Get resources from the host app, e.g., Facebook

Mini-app paradigm popularity:
- # of Host app downloads: 2,600,000,000 +
- # of Mini-app: 1,000,000+

New Security and Privacy Challenges

- No prior works provide full, necessary knowledge to systematize the policies for managing diverse systems resources on mobile platforms.
- Constructing sound security policies to govern system resources provided to mini-apps is difficult due to knowledge gap between host app developer and OS-level resource management.
- OS-level security policies are often opaque, inaccessible, and even misleading.
- No Prior techniques could properly identify restricted data in mini-apps systems, track for privacy leaks and audit non-compliance/regulation issues.

Vulnerability #1: System Resource Exposure

Inconsistent protection between System API and Sub-app API

System Resources

- WiFiManager requires ACCESS_FINE_LOCATION
- NEHotspotHelper requires HOTSPOT_HELPER

Root Causes
- Why would WifiManager require Location?
- Does HotspotHelper also safeguard location?
- What should the wx.getWifiList require?

Affected Host Apps

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>iOS</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Resource Exposure</td>
<td>A, D, J, Q, T, W</td>
<td>A, D, J, Q, T, W</td>
</tr>
<tr>
<td>Sub-window Deception</td>
<td>A, B, D, W, S, Q</td>
<td>A, B, C, D, F, W, O, Q</td>
</tr>
<tr>
<td>Sub-app Lifecycle Hijacking</td>
<td>---------------</td>
<td>A, W, Q</td>
</tr>
</tbody>
</table>

A: Alipay; B: Baidu; C: Chrome; D: DingTalk; F: Firefox; J: JinRiTouTiao; O: Opera; Q: QQ; S: Safari; T: TikTok; W: WeChat

Future Works

Security Gaps:
Systematic investigation is needed to understand the new challenges and attack vectors unique to the mini-app paradigm.

Privacy Compliance:
Privacy compliance analysis will focus on the gap between the convoluted app-in-app data practices and existing privacy model/regulations.

Defense:
Construct sound mini-app level security policies; Standardize mini-app interfaces and authorization frameworks.

**Title:** Demystifying Resource Management Risks in Emerging Mobile App-in-App Ecosystems

**Authors:** Haoran Lu, Luyi Xing, Yue Xiao, Yifan Zhang, Xiaoqing Liao, Xiaofeng Wang, Xueqiang Wang

**Email:** {haorlu, luyixing, xiaoyue, yz113, xliao, xw7, xw48}@indiana.edu

**Date:** November 9-13, 2020

**Venue:** Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS’20)

**DOI:** https://dl.acm.org/doi/10.1145/3372297.3417255

**ABSTRACT**

App-in-app is a new and trending mobile computing paradigm in which native app-like software modules, called sub-apps, are hosted by popular mobile apps such as Wechat, Baidu, TikTok and Chrome, to enrich the host app's functionalities and to form an “all-in-one app” ecosystem. Sub-apps access system resources through the host, and their functionalities come close to regular mobile apps (taking photos, recording voices, banking, shopping, etc.). Less clear, however, is whether the host app, typically a third-party app, is capable of securely managing sub-apps and their access to system resources. In this paper, we report the first systematic study on the resource management in app-in-app systems. Our study reveals high-impact security flaws, which allow the adversary to stealthily escalate privilege (e.g., accessing the camera, photo gallery, microphone, etc.) or acquire sensitive data (e.g., location, passwords of Amazon, Google, etc.). To understand the impacts of those flaws, we developed an analysis tool that automatically assesses 11 popular app-in-app platforms on both Android and iOS. Our results brought to light the prevalence of the security flaws. We further discuss the lessons learned and propose mitigation strategies.