Towards Discovering and Understanding Task Hijacking in Android

CHUANGANG REN, YULONG ZHANG, HUI XUE,
TAO WEI, PENG LIU

PENN STATE UNIVERSITY, FIREEYE INC.
In PC world, multitasking means multiple processes are running at the same period of time.

In Android, multitasking is a different concept:

“A task is a collection of activities that users interact with when performing a certain job”

- Android developer documentation
Android Multi-tasking

Activity Manager Service

Foreground

Task 1
- C
- B
- A

Task 2
- B
- A

Task 3
- C
- B
- A

Back Stack

PENN State
Android Multi-tasking

- Task switching
- UI navigation
- Task customization
However, the security implication of Android multitasking remains under-investigated

- Android allows activities from different apps to reside in the same task (or back stack)
- Android offers developers great flexibility to customize task behaviors

We find that Android multitasking is plagued by a serious security risk – \textit{task hijacking}
Example - User Spoofing
User Spoofing Attack

Normal Case

$S_1$
Foreground
Home Launcher

$S_2$
Foreground
Home Launcher Paypal

Attack Case

$S_1'$
Foreground
Home M2 M1 Launcher Malware

$S_2'$
Foreground
Home M1 M2 P1 Launcher Malware Paypal
How does mal-activity migrate?

The malware tricks the system to relocate the malicious activity (M2) to the Paypal task by manipulating the following task control knobs:

- Task affinity
- allowTaskReparenting
Task Affinity

- An activity attribute defined in each `<activity>` tag in `AndroidManifest.xml`

- Task affinity specifies which task that the activity desires to join. By default, all activities in an app have the same affinity – the app package name

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.app" >

  <application>
    <activity android:name=".ActivityA" />
    <activity android:name=".ActivityB" android:taskAffinity="com.example.app:taskB"/>
  </application>

</manifest>
```
Task Affinity

- Developer can re-define the task affinity in order to achieve desirable task behavior
  - Group activities into different tasks
  - Place activities defined in different apps within the same task

- If `<allowTaskReparenting = "true">` for activity A, and when a task with the same affinity as A is brought to the front, the system would move the “relocatable” activity A from its original hosting task to this new foreground task
User Spoofing Attack

Malware abuses the following task control knobs:

1. Activity M2: `taskAffinity = com.paypal.android`
2. Activity M2: `allowTaskReparenting = true`
Research Questions

- Question 1: How many types of task hijacking?
- Question 2: How to craft the individual attacks?
- Question 3: How to assess the vulnerability?
- Question 4: How to defend task hijacking?
Task Control Knobs

- We find that there are a rich set of task control knobs that can be abused by a task hijacking attack.
- Task control knobs in 4 categories:

<table>
<thead>
<tr>
<th>Intent Flag</th>
<th>Activity Attribute</th>
<th>Call-back Function</th>
<th>Framework API</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW_TASK</td>
<td>launchMode, taskAffinity, allowTaskReparenting, documentLaunchMode, FinishOnTaskLaunch</td>
<td>onBackPressed()</td>
<td>TaskStackBuilder class startActivity() startActivities()</td>
</tr>
<tr>
<td>SINGLE_TOP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REORDER_TO_FRONT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO_HISTORY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR_TASK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEW_DOCUMENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULTIPLE_TASKS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Methodology

- We formalize the task dynamic as a state transition model
  - **Hijacked task state**: at least one task in the system contains both malicious and benign activities
  - **Hijack state transition (HST)**: state transition that leads the system to a hijacked task state

- We simulate an Android system with three apps
  - Two benign apps (A, B), one malware (M)
  - Connect task states and generate task state transition graph
  - Flag the hijacked task states and HST in the graph
Two types of Hijacking State Transitions (HST):

- Malware activity moves to benign app task
- Benign activity is placed into malware task
Observations:

- There are many possible hijacking state transitions (HSTs)
- Once exploited, the HSTs could result in practical and serious real-world attacks
### Question 2 – Enabled attacks

We implemented 6 proof-of-concept attacks in 3 categories:

<table>
<thead>
<tr>
<th>Attack Category</th>
<th>Consequence</th>
<th>Attack Name</th>
<th>Vulnerable Systems &amp; Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Spoofing</td>
<td>Sensitive information stolen</td>
<td>Spoofing attack</td>
<td>all ; all</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Phishing attack (I–III)</td>
<td>all ; some apps</td>
</tr>
<tr>
<td>Denial-of-service</td>
<td>App function disabled; Restriction of user access</td>
<td>Ransomware</td>
<td>&gt;Android 5.0 ; all</td>
</tr>
<tr>
<td>User Monitoring</td>
<td>User privacy infringement</td>
<td>Spyware</td>
<td>&gt;Android 5.0 ; all</td>
</tr>
</tbody>
</table>

Task hijacking attacks affect all latest Android versions and apps, including the most privileged apps!
Question 3: Vulnerability Assessment

- We would like to first understand the use of security-sensitive task control knobs in real implementation.
- We analyze 6.8 million apps from Google Play and other 12 popular third-party app markets.

<table>
<thead>
<tr>
<th>Activity Attribute</th>
<th>% of Apps</th>
<th>Intent Flag</th>
<th>% of Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowTaskReparenting=&quot;true&quot;</td>
<td>0.80</td>
<td>NEW_TASK</td>
<td>79.42</td>
</tr>
<tr>
<td>launchMode=&quot;singleTask&quot;</td>
<td>24.63</td>
<td>CLEAR_TOP</td>
<td>37.59</td>
</tr>
<tr>
<td>launchMode= other non-default modes</td>
<td>24.75</td>
<td>EXCLUDE_FROM_RECENTS</td>
<td>10.08</td>
</tr>
<tr>
<td>taskAffinity= own pck. name</td>
<td>2.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskAffinity= other</td>
<td>1.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>excludeFromRecents=&quot;true&quot;</td>
<td>12.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alwaysRetainTaskState=&quot;true&quot;</td>
<td>2.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Events</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onBackPressed()</td>
<td></td>
<td></td>
<td>62.00</td>
</tr>
<tr>
<td>TaskStackBuilder</td>
<td></td>
<td></td>
<td>7.27</td>
</tr>
<tr>
<td>startActivities()</td>
<td></td>
<td></td>
<td>5.47</td>
</tr>
</tbody>
</table>
Case Study – Task Affinity

- 1.6% (109K apps) of all apps set the task affinity without containing their own package name
- These apps may interfere with the multitasking behaviors of other apps
  - *Unintentional*: careless app developers who are unaware of the security implications.
  - *Intentional*: task affinity intentionally set to popular app’s package name in order to implement legitimate “add-on” feature for these popular apps.
- We have not found evidence that malware has already abused these task control knobs
Question 4: Defense Suggestions

Detection in app review process
- App review guideline may contradict with existing app features
- Challenging to detect stealthy dynamic behaviors of an advanced malware

More secure multi-tasking mechanism
- Introduce additional security features for multitasking control
- For example, task affinity should comply with certain name space specification
- Introduce additional Boolean attribute to control if the app allow other apps to specify the same task affinity
Phishing attack
- A malware can steal user Citi Bank account name and password by hijacking citi bank task with a spoofing Citibank login interface

Denial of service
- A malware can disable app uninstallation in a system
- The similar attack approach could be used to create a ransomware
Thank you!