Checking More and Alerting Less: Detecting Privacy Leakages via Enhanced Data-flow Analysis and Peer Voting

Kangjie Lu, Zhichun Li, Vasileios P. Kemerlis, Zhenyu Wu, Long Lu, Cong Zheng, Zhiyun Qian, Wenke Lee, Guofei Jiang
In terms of privacy data...

• Are “you” contained in smartphone?
  – Contacts, photo, SMS, credentials, browse history...
Privacy Disclosures

• Privacy can be disclosed to internet or public
Prevalent Privacy Disclosures

– **8%** apps failed to protect bank account and social media logins \[\text{BBC, Oct 12}\]

– **95%** of the top-100 free exhibited at least one kind of privacy-compromising behavior, while **78%** of paid apps disclosed similar data. \[\text{Black Hat USA, Jul 13}\]

– **30%** general apps have privacy disclosures, shown by AndroidLeaks \[\text{TRUST’12}\]
Privacy Disclosure Vs. Privacy Leak

• Privacy Disclosure == Privacy Leak?
• MOST privacy disclosures are legitimate
Research Problem

• How can we automatically differentiate suspicious privacy leaks from legitimate privacy disclosures???
Insight

• An app’s (namely primacy app) functionally similar apps (namely peer apps) are supposed to exhibit similar privacy disclosures

• If a privacy disclosure is uncommon in peer apps → likely suspicious
AAPL: Analysis of App Privacy Leak

Detecting Privacy Leaks via Peer Voting Mechanism
AAPI Workflow

1. Collect Peer Apps
2. Uncover Privacy Disclosures
3. Detect Leaks with Peer Voting

Privacy & Peer Apps

Privacy Disclosures

Privacy Leak Report (Output)

Primary App (Input)

AAPI

10/2/15
Collecting Peer Apps

• Possible approaches
  – Apps with similar permissions 😞
  – Apps with similar text descriptions 😞
  – Similar apps suggested by Google Play,
    • derived from users’ experience, ML, etc. 🙂
Purifying Peer Apps

• Adopt NLP
  – Parse app descriptions using NLTK\textsuperscript{1}
  – tf-idf vectors
  – cosine similarity

\textsuperscript{1}http://www.nltk.org/
Uncovering Privacy Disclosures
Opportunistic Constant Evaluation

• Conditional sources
  • `provider.query(uri, sql)`
    • Contact? SMS?
    • Non-sensitive?
      *Have no idea?*

• Backward SDG slicing & DFS traversal

```java
Uri uri = "1";
...
uri = uri.concat("2");
...
Data = provider.query(uri);
```

Most privacy data accesses follow this way
Peer Voting

• **VotesNumber**: the total number of peers with the same disclosure (votes with 1)
• **PeersNumber**: the number of peers
• **Disclosure legitimacy** = VotesNumber/PeersNumber
Implementation

• Built on Dalysis[^CHEX CCS’12] and IBM WALA^1

• The improvements account for about 6K SLoC in Java; Peer voting accounts for 1.3K SLoC in Python

[^1]: http://wala.sourceforge.net/
Evaluating Disclosure Analysis

• Data set: 40,456 apps; manually examined 530 data-flows in top 300 popular apps

• Performance: 12 seconds/app

• Detection rate: 44.7% (31% increased compared with original 36.9%)

• False positive rate: 6.7% (5 times reduced compared with original 34.2%)
Evaluating Peer Voting

• Manually label 532 unique privacy disclosures from 417 randomly chosen primary apps

Privacy Disclosures

- 67% Legitimate Privacy Disclosure
- 33% Privacy Leak

• Accuracy: 88.7% with false positive rate 10.7% and false negative rate 12.5%
## Case Studies

<table>
<thead>
<tr>
<th>App ID</th>
<th>Leak</th>
<th># of peers</th>
<th>Legitimacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.linpusimetc.android.linpustckbd</td>
<td>Contacts -&gt; URL</td>
<td>20</td>
<td>0%</td>
</tr>
<tr>
<td>simosoftprojects.musicplayerforpad</td>
<td>Phone Number -&gt; URL</td>
<td>21</td>
<td>0%</td>
</tr>
<tr>
<td>com.apptivateme.next.hr dp</td>
<td>Cookie -&gt; Log</td>
<td>15</td>
<td>0%</td>
</tr>
</tbody>
</table>
Conclusion

• We propose AAPL, a novel peer voting mechanism to detect suspicious privacy leaks
• Checking more and alerting less
Thank you!

Q & A