Panoptispy: Characterizing Audio and Video Exfiltration from Android Applications

Elleen Pan, Jingjing Ren, Martina Lindorfer*, Christo Wilson, and David Choffnes

Northeastern University, *UC Santa Barbara
Motivation

+ internet connectivity … 🕵️‍♂️ 🕵️‍♀️
Examples

ultrasonic beacons for cross-device linking
patents for recognizing user emotion
listening for unlicensed broadcasting
photos taken surreptitiously by shrinking preview to 1x1 pixel

Media surveillance, so far, has been anecdotal
Goals

• Identify & measure media (audio, images, video) exfiltration **at scale**
  • Large number of apps & broad coverage of app stores
• Focus on exfiltration over network
• Is the exfiltration a **leak** (undisclosed/unexpected)?

• How do apps use sensors?
  • Permissions requested
  • APIs called
  • First or third-parties
Outline

• Motivation
• Threat Model
• Methodology
• Aggregate Results
• Case Studies
  • Photography apps
  • Screen recording
• Discussion
• Conclusion
Android Access Control

• Certain APIs require permissions in order for code to execute
• Protects sensors from being accessed by apps that don’t need it
• Requested at install time for API level 22-, runtime for API level 23+
Android Permission Model

- Camera & mic hardware access
Why aren’t permissions enough?

• Incomplete
  • No permissions required for capturing app screen

• Coarse-grained
  • Permissions granted at app level
    • Third-party libraries also get access
    • Users don’t know when apps are using hardware

• Lack of visibility and control (may contain PII!)
  • as media is exfiltrated over the network
  • Background access
Definition of media leak

Suspicious or unexpected

1. Does it further the primary purpose of the app?
2. Is it disclosed to the user?
   • Privacy policies
3. Is it employed by similar apps?
4. Is it encrypted over the internet?

No? It’s a leak
App Selection

Popular + new from **Google Play**

Popular + random from **AppChina, Mi.com, Anzhi**

Camera or audio permission

<table>
<thead>
<tr>
<th>Store</th>
<th># of apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Play</td>
<td>15,627</td>
</tr>
<tr>
<td>AppChina</td>
<td>510</td>
</tr>
<tr>
<td>Mi.com</td>
<td>528</td>
</tr>
<tr>
<td>Anzhi</td>
<td>285</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17,260</strong></td>
</tr>
</tbody>
</table>
Static Analysis

- Permission analysis (camera, record audio)
- Media API references (camera, record audio, video, screen capturing)
- Media API references found in third-party libraries
Dynamic Analysis

• Why is dynamic analysis necessary?
  • Detect whether media permissions are actually used
  • Media APIs may be in dead code paths
  • Detect dynamically loaded / obfuscated code
Dynamic Analysis

• Test environment
• Automated interaction
  • Monkey for 5,000 events
• Recording network traffic
  • Mitmproxy to intercept traffic
Detection of Media in Network Traffic

- **Extraction**
  - Mediaextract detection with **file magic numbers**
    - E.g. JPEG files: FF D8 FF …
    - False positives require manual checking

<table>
<thead>
<tr>
<th>Category</th>
<th>Supported</th>
<th>Unsupported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>3gp, aac, id3v2, m4a, ogg, wav</td>
<td>raw</td>
</tr>
<tr>
<td>Image</td>
<td>bmp, gif, jpg, png, webp</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>3gp, mp4, webm</td>
<td></td>
</tr>
</tbody>
</table>
Detection of Media in Network Traffic

• Validation
  • Test app
  • Manual tests with known apps – imgur, GIPHY, SoundCloud
  • Verification of detected media by manually interacting with apps
Static: Permission vs. API

- Large fractions of audio (43.8%) and camera (75.6%) permission declarations
- Permissions > API calls
- Mi, Google > Anzhi, AppChina
- One exception: API > permission (audio in Play)
Dynamic: Media in Network Traffic

- 21 cases of detected media – 12 considered leaks
  - Unexpected or unencrypted
- 9 shared with third parties
Case Study: Photography Apps

• Server-side photo editing
  • Photos are sent to servers
  • Users not notified

• App has no other functionality requiring internet connection

• Privacy policy vaguely disclosed (5 apps) or didn’t mention (1 app)
Case Study: Screen Recording

goPuff
• Screen recording of user interaction, where PII was exposed
  • Leaked to an Appsee domain

Appsee
• Screen recording as a feature
• Developers are responsible for hiding sensitive screens
• Few apps use the API method to do so – 5/33 apps
  • Server-side way exists, unknown how many apps use it
Responsible Disclosure

• Pulled Appsee from Android & iOS builds
• Updated privacy policy

Google

• Reviewed GoPuff & Appsee
  • “Google constantly monitors apps and analytics providers to ensure they are policy-compliant. When notified of our findings, they reviewed GoPuff and Appsee and took the appropriate actions.”
Limitations

• Translated media formats (audio being transcribed, etc.)
• Controlled experiments do not replicate environmental conditions
• Intentional obfuscation of traffic
These Academics Spent the Last Year Testing Whether Your Phone Is Secretly Listening to You

Your phone isn’t listening to you, researchers say, but it may be watching everything you do.

No, your smartphone is not listening to your conversations, but they do something equally creepy.

Smartphone apps don’t listen to your conversations, but they do something equally creepy.

They found that while smartphone applications did not send audio clippings to third-party domains, they did send screenshots or screen recordings to them.
Recommendations

• Access to the screen should be protected by OS
  • Or, users should at least be notified & able to opt out

• Main app & third-party permissions should be separated
Conclusion

• Most apps have over-provisioned permissions
  • Susceptible for abuse from third parties
• 12 cases of unexpected or unencrypted media
  • 9 cases of third party sharing
• Screen recording video sent to a third party library
  • Sensitive input fields
  • No permissions or notification to the user

https://recon.meddle.mobi/panoptispy/
Threat Model

• Goal: media exfiltration from Android apps over the network

• Permissions
  • Not granted
  • Granted for a user-identifiable purpose

•Leaks: unexpected or suspicious
Experiment Design
## Permissions and API references

<table>
<thead>
<tr>
<th>Store</th>
<th># of Apps</th>
<th>Audio Permission</th>
<th>Audio API</th>
<th>Camera Permission</th>
<th>Camera API</th>
<th>Screenshot API</th>
<th>Screen recording API</th>
<th>External Storage Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anzhi</td>
<td>883</td>
<td>12.8%</td>
<td>9.7%</td>
<td>15.7%</td>
<td>11.7%</td>
<td>20.7%</td>
<td>1.5%</td>
<td>23.4%</td>
</tr>
<tr>
<td>AppChina</td>
<td>468</td>
<td>28.4%</td>
<td>22.9%</td>
<td>37.0%</td>
<td>28.6%</td>
<td>57.1%</td>
<td>2.4%</td>
<td>94.0%</td>
</tr>
<tr>
<td>Mi.com</td>
<td>392</td>
<td>55.9%</td>
<td>41.8%</td>
<td>61.0%</td>
<td>45.7%</td>
<td>81.6%</td>
<td>5.6%</td>
<td>97.4%</td>
</tr>
<tr>
<td>Google Play</td>
<td>15,627</td>
<td>45.7%</td>
<td>46.2%</td>
<td>80.5%</td>
<td>75.1%</td>
<td>89.1%</td>
<td>10.6%</td>
<td>92.7%</td>
</tr>
<tr>
<td>All</td>
<td>17,260</td>
<td>43.8%</td>
<td>43.6%</td>
<td>75.6%</td>
<td>70.1%</td>
<td>84.6%</td>
<td>9.8%</td>
<td>89.9%</td>
</tr>
</tbody>
</table>
Permissions and API references

- Large percentages of apps request media permissions
  - Smaller percentage actually call methods that use those permissions
- Multipurpose APIs for screenshots and accessing external storage
  - High false positive rate
- Nontrivial inconsistency between permissions and API calls
Static: API References
March 26
- Initial disclosure to GoPuff

March 27
- Lawyer contacts NEU and accuses us of extortion
- No direct reply to our team

March 29
- After some back and forth, updated privacy policy – by removing it?

May 15
- Notified GoPuff of absent privacy policy

June 7
- Informed that the lawyer is no longer with company, but introduced to CTO
- Start talking about Appsee & the screen recording

June 21
- GoPuff pulls Appsee from iOS & Android builds and updates their privacy policy
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 26</td>
<td>Initial disclosure to GoPuff</td>
</tr>
<tr>
<td>March 27</td>
<td>Lawyer contacts NEU and accuses us of extortion</td>
</tr>
<tr>
<td>March 27</td>
<td>No direct reply to our team</td>
</tr>
<tr>
<td>March 29</td>
<td>After some back and forth, updated privacy policy – by removing it?</td>
</tr>
<tr>
<td>May 15</td>
<td>Notified GoPuff of absent privacy policy</td>
</tr>
<tr>
<td>June 7</td>
<td>Informed that the lawyer is no longer with company, but introduced to CTO</td>
</tr>
<tr>
<td>June 7</td>
<td>Start talking about Appsee &amp; the screen recording</td>
</tr>
<tr>
<td>June 21</td>
<td>GoPuff pulls Appsee from iOS &amp; Android builds and updates their privacy policy</td>
</tr>
</tbody>
</table>
• "verges on defamation"
• Provided info about screen recording
  • didn’t have to do with privacy concerns
• Asked us to remove Appsee / screen recording
• We replied to their points and clarified the privacy risk
• No reply
• First reported as a security vulnerability
• Passed to privacy team
• “Google constantly monitors apps and analytics providers to ensure they are policy-compliant. When notified of our findings, they reviewed GoPuff and Appsee and took the appropriate actions.”
Screen Capturing

• Testfairy
  • Screenshots of app while in use
  • Library intended for beta testing
    • App was not a beta version in the Google Play store
    • User not informed of recording, not given a prompt to consent to beta testing