Leave Me Alone: App-level Protection Against Runtime Information Gathering on Android

Nan Zhang, Kan Yuan, Muhammad Naveed, Xiaoyong Zhou and XiaoFeng Wang

Presented by Hitakshi Annayya
Contents

1. RIG Attacks
2. Android-Based IoT
3. Previous Works
4. App Guardian
5. Evaluation and Analysis
6. Conclusion
7. References
RIG Attacks

Runtime-Information-Gathering (RIG)

- Collect runtime information from target app (the victim)
- Directly steal or indirectly infer sensitive user information

1) Design weaknesses of the OS
   shared communication channels such as Bluetooth
2) Side channels
   memory and network-data usages
Android Permission Issues

A malicious app needs to run side-by-side with the target app (the victim) to collect its runtime information.

A malicious app can abuse the permission it gets “to directly collect sensitive data from the target app running in the foreground.”

RIG attacks exploit apps to obtain sensitive user data “ranging from phone conversations to health information;”

A game app with the Bluetooth permission for connecting to its playpad can also download patient data from a Bluetooth glucose meter.”
Android-based Internet of Things (IoT)

1. Belkin NetCam Wi-Fi Camera with Night Vision
   Designed for home surveillance and motion detection
   Report to the house owner remotely

2. Nest Protect
   Shipped 440,000 of its smoke alarms in the United States between Nov. 2013 and Apr. 2014
NetCam Communication Model
NetCam Attacks

- Utilize two side channels
  - Traffic statistics: tcp_snd and tcp_rcv
  - CPU usage: /proc/<pid>/stat

Three steps
- Infer if anybody is at home
- Mute alarm
- Infer anybody is watching surveillance

https://sites.google.com/site/appguardian/
How to Protect from RIG attack?
Previous Works

- **Enhancing access control causes compatibility issues**
  - Prevent information leaks during security-critical operations such as phone calls
  - Remove public resources that could be used for a side-channel analysis
  - Inevitably make the system less usable
  - Cause compatibility issues
Previous Works

- **Modify OS**

  Complicated and painful (Android OS ecosystem: fragmentation)

  - New protection takes a long time before it can reach Android devices worldwide;
  - New RIG attacks continue to be brought to the spotlight;
  - It is less clear what an app can do by itself to control its information exposed by the OS.
Researchers proposed solution

App Guardian
1. Information Gathering
   - Permissions, side-channels
2. Install / Run time features
3. Report suspicious apps
4. Kill suspicious app
5. Principal finished
6. Resume suspicious app
Grant Guardian a set of permissions

- KILLBACKGROUND_PROCESSES – for closing other third-party apps
- SYSTEM ALERT WINDOW - for popping up an alert to the user
- INTERNET – to access internet
- GET TASK - for getting top activity
- BIND_NOTIFICATION_LISTENER_SERVICE - for controlling notifications
Life cycle of Guardian Protection

Normal Mode

Ward Mode
Monitoring

action.PACKAGE_ADDED

package names, permissions...

proc files, recording thread …
Entering the ward

WARD MODE

BACKGROUND APPS

KILL_BACKGROUND_PROCESSES
Entering the ward

oom_adj score (-17 ~ 15)

(typically) 9

2
Exiting the ward
Impacts on Performance

• Close an app which might be restarted later
  + App states are well preserved
  - Take longer time than Switch to foreground

<table>
<thead>
<tr>
<th>App</th>
<th>Restart (s)</th>
<th>Switch (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subway Surf</td>
<td>9.76</td>
<td>2.89</td>
</tr>
<tr>
<td>Mx Player</td>
<td>1.15</td>
<td>0.55</td>
</tr>
<tr>
<td>Flashlight</td>
<td>1.27</td>
<td>0.68</td>
</tr>
<tr>
<td>Shazam</td>
<td>2.18</td>
<td>0.77</td>
</tr>
<tr>
<td>RunKeeper</td>
<td>4.02</td>
<td>1.35</td>
</tr>
<tr>
<td>Bible.is</td>
<td>2.47</td>
<td>0.58</td>
</tr>
<tr>
<td>Chase</td>
<td>1.94</td>
<td>0.75</td>
</tr>
<tr>
<td>Duolingo</td>
<td>2.92</td>
<td>0.95</td>
</tr>
<tr>
<td>PicsArt</td>
<td>2.08</td>
<td>0.91</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>1.91</td>
<td>0.65</td>
</tr>
</tbody>
</table>
Finding suspicious App

- Use malicious app’s side channel
Finding suspicious App (Cont.)

Data Stealing Attacks

1. RECORD_AUDIO permission
2. Start Audioin_X process to record audio (/proc/<pid>/task/<tid>/status)

Side-channel Attacks

- How frequently app uses the CPU resources
- Number of times schedule to use CPU
Behavior change

• Challenge:
  - keep low profile before the principal show up
  - act aggressively afterwards

• Solution:
  Pearson correlation coefficient (r)

<table>
<thead>
<tr>
<th>α</th>
<th>1 − β</th>
<th>r</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>0.90</td>
<td>7</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>0.95</td>
<td>5</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>0.98</td>
<td>4</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>0.9993</td>
<td>3</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Collusion

• Challenge:

Multiple apps sample at a lower rate but still collect sufficient information

• Solution:

- Grouping apps with same signature
- Detect link-installed apps
- Ask user if less obvious recommendation
Self Protection

• Use `startForeground` to start a service
  Prevent it from killed by
  `KILL_BACKGROUND_PROCESSES`
Evaluation and analysis
Effectiveness

- Defeat all 12 RIG Attacks

<table>
<thead>
<tr>
<th>No.</th>
<th>RIG Attacks</th>
<th>Defeat</th>
<th>Attack Success Rate (SR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audio Recording</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Bluetooth Data Stealing</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>Alarm Blocking</td>
<td>Yes</td>
<td>Fail (2/s)</td>
</tr>
<tr>
<td>4</td>
<td>Motion Detection On</td>
<td>Yes</td>
<td>Fail (1/3s)</td>
</tr>
<tr>
<td>5</td>
<td>WebMD: inferring disease conditions</td>
<td>Yes</td>
<td>RG (1/2s)</td>
</tr>
<tr>
<td>6</td>
<td>Twitter: inferring identities</td>
<td>Yes</td>
<td>RG (end-to-end)</td>
</tr>
<tr>
<td>7</td>
<td>Web Page Inference</td>
<td>Yes</td>
<td>RG (10/s)</td>
</tr>
<tr>
<td>8</td>
<td>Driving Route Inference</td>
<td>Yes</td>
<td>Fail (1/s)</td>
</tr>
<tr>
<td>9</td>
<td>Keylogger 1: TouchLogger</td>
<td>Yes</td>
<td>$\leq 1/3s$ (1/3s)</td>
</tr>
<tr>
<td>10</td>
<td>Keylogger 2: Screenmilkler</td>
<td>Yes</td>
<td>$\leq 1/3s$ (1/3s)</td>
</tr>
<tr>
<td>11</td>
<td>Voice eavesdropping</td>
<td>Yes$^5$</td>
<td>Fail (1/3s)</td>
</tr>
<tr>
<td>12</td>
<td>UI inference</td>
<td>Yes$^5$</td>
<td>Fail (1/3s)</td>
</tr>
</tbody>
</table>
Utility Impacts and Performance

- 475 popular Apps from 27 categories on Google Play Store
  - 92 apps (19.3%) apps potentially needs to be closed
  - 8 apps (1.68%) may affect phone users’ experience

<table>
<thead>
<tr>
<th>App</th>
<th>Category</th>
<th>SR</th>
<th>oom_adj</th>
<th>Recoverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>Social</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Fox News</td>
<td>News &amp; Magazines</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Yelp</td>
<td>Travel &amp; Local</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Viber</td>
<td>Communication</td>
<td>1/1</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Amazon</td>
<td>Shopping</td>
<td>2/1</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>The Weather Channel</td>
<td>Weather</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>FIFA</td>
<td>Sports</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Temple Run 2</td>
<td>Games</td>
<td>10/1</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Photo Grid</td>
<td>Photography</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>Adobe Reader</td>
<td>Productivity</td>
<td>&lt; 1/3</td>
<td>9</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Overhead

• **CPU & Memory usage**
  Two Nexus5 phones with 250 apps installed on each
  - In ward mode, 5% CPU Resource, 40MB Memory
  - Out of ward mode, < 1% CPU

• **Battery Usage**
  Two Nexus5 phones with 50 apps installed on each
  - In ward mode, 0.12% ~ 0.18% per hour
  - Out of ward mode, 0.75% ~ 1.05% per day
  - Estimate a day, 0.84~ 1.18% per day
Discussion and future work

• Detection and Separation
  A more accurate identification of malicious activities will help

• Background process protection
  Protect background process at minimal cost

• Sanitization
  Thoroughly clean up the principals’ execution environment after the program stop running

• Possible side-channel attack on iOS / WatchOS
Conclusion

- Serious of RIG attacks on Android
  IoT systems are also vulnerable

- App Guardian
  - App level protection
  - Uses side channel to protect principle
Thank you!