AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings

Giuseppe Petracca
gxp18@cse.psu.edu
The Pennsylvania State University
School of Electrical Engineering and Computer Science
Institute for Networking and Security Research

Ahmad-Atamli Reineh
atamli@cs.ox.ac.uk
University of Oxford, UK
Dept. of Electrical Engineering and Computer Science

Yuqiong Sun
yuqiong_sun@symantec.com
Symantec Research Labs, US

Jens Grossklags
jens.grossklags@in.tum.de
Technical University of Munich, DE

Trent Jaeger
tjaege@cse.psu.edu
The Pennsylvania State University
School of Electrical Engineering and Computer Science
Institute for Networking and Security Research
Controlling when **applications** may use **privacy-sensitive sensors** (i.e., cameras, microphones, and touch screens):
Abuse of Privacy-Sensitive Sensors

G. Petracca et al. - AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings

PlaceRaider: Virtual Theft in Physical Spaces with Smartphones
Robert Templeman, Zahid Rahman, David Crandall, Apu Kapadia

Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones.
Schlegel, Roman and Zhang, Xchuan and Zhoa, Xio-yong and Intwala, Mchool and Kapadia, Apu and Wang, XioFeng
Symantec discovered a new HTTP Android Remote administration tool, named Dendroid, available on the underground market for only $300.

$610K Settlement in School Webcam Spy Case

Last February, the Lower Merion School District outside Philadelphia came under fire for using laptop webcams to look in on students at home. Last week, the school district settled legal action stemming from those actions.

Krysanec trojan: Android backdoor lurking inside legitimate apps

BY ROBERT LIPOVSKY POSTED 12 AUG 2014 - 12:21PM

Lawsuit claims popular Warriors app accesses phone’s microphone to eavesdrop on you

By Kate Dowd | SFGATE | Updated 3:11 pm, Thursday, September 5, 2019

SAN FRANCISCO — Want to invisibly spy on iPhone owners without their knowledge? Gather their every keystroke, sound, message and location? That will cost you $650,000, plus a $500,000 setup fee with an Israeli outfit called the NSO Group. You can spy on more people if you would like — just check out the company’s price list.

FTC Issues Warning Letters to App Developers Using ‘Silverpush’ Code

Letters Warn Companies of Privacy Risks In Audio Monitoring Technology
Beginning in Android 6.0 (API level 23), users grant permissions to apps while the app is running, not when they install the app!
Shortcomings

G. Petracca et al. - AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings
Shortcomings

First-Use  
Install-Time  
Install  
Use  
Use  

G. Petracca et al. - AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings
Proposed Defenses

Input-Driven Access Control (IDAC)
Authorize an operation request that immediately follows a user input event

User inputs associated with operation authorizations
Binding between the user inputs and the authorized operations still unknown to the system!
Proposed Defenses

User-Driven Access Control (UDAC)
Applications must use **system-defined gadgets** associated with particular operations

Binding between the user input and the authorized operation explicit to the system

Binding still not explicit to the user!
User-Driven Access Control (UDAC)

Applications must use system-defined gadgets associated with particular operations

Compatibility Issue
User-Driven Access Control (UDAC)
Applications must use system-defined gadgets associated with particular operations

300,000+ apps Need Redesign

Audacious: User-driven access control with unmodified operating systems
T. Ringer, D. Grossman, F. Roosner
Proceedings of the 2016 ACM ..., 2016 - dl.acm.org

User-driven access control: Rethinking permission granting in modern operating systems
F. Roosner, T. Kohno, A. Moischuk
Security and privacy ..., 2012 - ieexplore.ieee.org
Limitations of Prior Work

Leverage the user as weak link to circumvent protection mechanisms!

“User Interface Attacks”

User may fail to:

- Identify the application requesting sensor access
- Recognize subtle changes in the Graphic User Interface (GUI)
- Understand the operation granted by a particular gadget
Use Interface Attacks (Bait-and-Switch)

Interaction #1
Use Interface Attacks (Bait-and-Switch)

Interaction #2
Use Interface Attacks (*Bait-and-Switch*)

Interaction #3
Use Interface Attacks (*Bait-and-Switch*)

Interaction #4
Use Interface Attacks (Bait-and-Switch)

Interaction #5
Use Interface Attacks (Bait-and-Switch)

Interaction #4

The application maintained the windowing display context but switched the widget to record audio.

“Bait-and-Switch Attack”
Use Interface Attacks (Application Spoofing)

A click by the user allows the Legitimate App to record audio.
Use Interface Attacks (Application Spoofing)

A click by the user allows the Spoofing App to record audio

“Application Spoofing Attack”
Research Objectives

- **Operation authorizations explicit to both the system and the user**
- **Compatible with pre-existing applications**
- **Low authorization effort for the user (~ First-Use)**
- **No perceivable performance overhead**

**AWARE**: Authorization Framework extending OS middleware to make access to privacy-sensitive sensors **explicit** to both the system and the user
**Goal:** Identify the app’s widgets available to the user for requesting access to privacy-sensitive sensors
Challenge: Identify legitimate entry points (Widgets)

**Insight:** Bind each user input even with the application’s widget displayed on the screen

**Operation Binding**

\[(app, e, w)\]

- \(app\) = application associated with widget
- \(e\) = user input event
- \(w\) = user interface widget
**Goal**: Identify the particular user interface configuration within which each widget is allowed to appear on the screen.
Insight: Bind the widget with a set of structural features that uniquely identify the UI configuration

**Operation Binding**

\((\text{app}, \ e, \ w, \ c)\)

- \(\text{app}\) = application associated with widget
- \(e\) = user input event
- \(w\) = user interface widget
- \(c\) = user interface configuration containing the widget
Goal: Make the application’s requested operation and the target sensors explicit to the system.
Insight: Mediate each application’s operation request and identify the privacy-sensitive sensors targeted by such operation. Bind the application’s request to a specific input event for a particular widget.

Operation Binding

\((app, S, op, e, w, c)\)

- \(app\) = application associated with widget and operation request
- \(S\) = set of sensors targeted by the request
- \(op\) = operation being requested
- \(e\) = user input event
- \(w\) = user interface widget
- \(c\) = user interface configuration containing the widget
Challenge: Make the Operation Binding explicit to the user

Currently
(First-Use)
AWARE’s Explicit Binding Request

(app, S, op, e, w, c)
are now explicit!
AWARE’s Explicit Binding Request

(app, S, op, e, w, c) are now explicit!
AWARE’s Explicit Binding Request

(app, S, op, e, w, c)
are now explicit!
AWARE’s Explicit Binding Request

Allow Instagram to use the front Camera to take Pictures when pressing?

Allow Deny

(e) (Input Event)

(app, S, op, e, w, c) are now explicit!
AWARE’s Explicit Binding Request

(app, S, op, e, w, c) are now explicit!
AWARE’s Explicit Binding Request

(app, S, op, e, w, c) are now explicit!
Challenge: Limit User Effort

**Goal**: Limit the number of *Explicit Authorization* required by the User!
Challenge: Limit User Effort

Insight: Cache Operation Bindings!

A user’s authorization of an operation binding implies that:

“The application will be allowed to perform the requested operation on the set of sensors whenever the user produces the same input event using the same widget within the same user interface configuration”
Preventing Bait-and-Switch Attacks

Operation Binding
\((app, S, op, e, w, c)\)

Display Context
\(c = \text{set of structural Features of the most enclosing activity window containing the widget } w\)
Preventing Application Spoofing Attacks

**Activity Window Call Graph**

Nodes represent activity windows

Edges represent enabled transitions

(user input or system events)
Preventing Application Spoofing Attacks

Security Message

Activity Window Call Graph
- Nodes represent activity windows
- Edges represent enabled transitions
- (user input or system events)

```
app, S, op, e, w, c
```

G. Petracca et al. - AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings
Experimental Evaluation

Prototyped (Android OS 6.0.1_r5)
Tested (Nexus 5 and Nexus 5X smartphones)

Research Questions:

- To what degree is the AWARE operation binding concept assisting the users in avoiding attacks? (Effectiveness)

- What is the decision overhead imposed to users due to per-configuration access control? (Usability)

- How many existing apps malfunctioned due to the integration of AWARE? (Compatibility)

- What is the performance overhead imposed by AWARE for the operation binding construction and enforcement? (Performance)
Effectiveness

To what degree is the AWARE operation binding concept assisting the users in avoiding attacks?

Laboratory-Based User Study (90 Participants)

Groups: Install-Time, First-Use, Input-Driven, System-Defined Gadgets, and AWARE
To what degree is the AWARE operation binding concept assisting the users in avoiding attacks?

Laboratory-Based User Study (90 Participants)

Groups: Install-Time, First-Use, Input-Driven, System-Defined Gadgets, and AWARE
Effectiveness

To what degree is the **AWARE** operation binding concept assisting the users in avoiding attacks?

Experimental Results:

- **TASK 1**: Operation performed by app not visible
  (Exception for Access Control Gadgets)
  Attack Prevention Rate: **Others 2% vs AWARE 100%**

- **TASK 2** and **TASK 3**: Users were successfully tricked by switching the user interface configuration!
  Attack Prevention Rate: **Others 2% vs AWARE 93%**

- **TASK 4**: Real identity of the app performing the operation was not visible to users
  Attack Prevention Rate: **Others 6% vs AWARE 100%**
Usability

What is the **decision overhead** imposed to users due to per-configuration access control?

Field-Based User Study (24 Participants)
- 21 apps (7 categories)*
- 1 week

(Comparison with First-Use)

4 explicit authorizations per-application on average

<table>
<thead>
<tr>
<th>App Category</th>
<th>App Name</th>
<th>Explicit User Authorizations</th>
<th>Total Operation Authorizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Recording</td>
<td>WhatsApp</td>
<td>3</td>
<td>1,217 (±187)</td>
</tr>
<tr>
<td></td>
<td>Viber</td>
<td>1 (±1)</td>
<td>88 (±9)</td>
</tr>
<tr>
<td></td>
<td>Messenger</td>
<td>7 (±2)</td>
<td>2,134 (±176)</td>
</tr>
<tr>
<td>Photo and Video</td>
<td>Facebook</td>
<td>2 (±1)</td>
<td>3,864 (±223)</td>
</tr>
<tr>
<td></td>
<td>SilentEye</td>
<td>5 (±1)</td>
<td>234 (±20)</td>
</tr>
<tr>
<td></td>
<td>Fideo</td>
<td>4 (±1)</td>
<td>213 (±23)</td>
</tr>
<tr>
<td>Screenshot Capture</td>
<td>Ok Screenshot</td>
<td>2 (±1)</td>
<td>49 (±8)</td>
</tr>
<tr>
<td></td>
<td>Screenshot Easy</td>
<td>1 (±1)</td>
<td>76 (±7)</td>
</tr>
<tr>
<td></td>
<td>Screenshot Capture</td>
<td>1 (±1)</td>
<td>64 (±4)</td>
</tr>
<tr>
<td>Screen Recording</td>
<td>REC Screen Recorder</td>
<td>2 (±1)</td>
<td>41 (±8)</td>
</tr>
<tr>
<td></td>
<td>AZ Screen Recorder</td>
<td>2 (±1)</td>
<td>49 (±7)</td>
</tr>
<tr>
<td></td>
<td>Rec.</td>
<td>3 (±1)</td>
<td>66 (±4)</td>
</tr>
<tr>
<td>Full Screen Mode</td>
<td>Instagram</td>
<td>6 (±1)</td>
<td>3,412 (±182)</td>
</tr>
<tr>
<td></td>
<td>Snapchat</td>
<td>6 (±1)</td>
<td>5,287 (±334)</td>
</tr>
<tr>
<td></td>
<td>Skype</td>
<td>9 (±3)</td>
<td>468 (±62)</td>
</tr>
<tr>
<td>Remote Control</td>
<td>Prey Anti Theft</td>
<td>8 (±2)</td>
<td>47 (±5)</td>
</tr>
<tr>
<td></td>
<td>Lost Android</td>
<td>6 (±1)</td>
<td>37 (±6)</td>
</tr>
<tr>
<td></td>
<td>Avast Anti-Theft</td>
<td>4 (±1)</td>
<td>34 (±7)</td>
</tr>
<tr>
<td>Hands-Free Control</td>
<td>Google Voice Search</td>
<td>1 (±1)</td>
<td>1,245 (±122)</td>
</tr>
<tr>
<td></td>
<td>HappyShutter</td>
<td>1 (±0)</td>
<td>3 (±1)</td>
</tr>
<tr>
<td></td>
<td>SnapClap</td>
<td>1 (±0)</td>
<td>4 (±2)</td>
</tr>
</tbody>
</table>

*www.statistica.com
Compatibility

How many existing apps malfunctioned due to the integration of AWARE?

Android Compatibility Test Suite (CTS):
- 1,000 apps (Google Play)
- 13 hours and 28 minutes

Experimental Results:
- 126,681 passed tests over 126,686
- [Viber] Camera and microphone probing at reboot (No impact on video or voice calls)
What is the performance overhead imposed by AWARE for the operation binding construction and enforcement?

Android UI/Application Exerciser Monkey:
- 1,000 app (Google Play)
- Nexus 5 and Nexus 5X

Microbenchmark:
- Access requests for operation targeting privacy-sensitive sensors
- 10,000 operations

Experimental Results:
- 0.33% system-wide performance overhead
- About 3 MB of cache for operation binding cache and window call graphs
Conclusion

- Authorization of sensor operations *explicit* to both system and user (*Operation Binding*)
  - Up to 100% user interface attack prevention (only up to 6% with alternative approaches)

- **Low user effort** (*Caching of Bindings* when UI is same for same operation)
  - 4 explicit authorizations per-application on average

- Compatible with pre-existing applications (*No app modification or redesign*)
  - Only 5 minor compatibility issues out of 1,000 tested apps

- **Negligible Performance Overhead** (limited number of authorization hooks and quick retrieval of bindings)
  - 0.33% performance overhead and 3 MB of cache
Thank You For Your Attention!

Giuseppe Petracca
Ph.D. Candidate
gxp18@cse.psu.edu
https://sites.psu.edu/petracca/

Source Code: https://github.com/gxp18/AWare
Approach Overview

**AWARE Authorization Workflow**

1. **User Interface (Widgets)**
2. **User Interface (ApplID, Widgets)**
3. **User Input (ApplID, Widget, Configuration)**
4. **User Input Events (ApplID, Widget, Configuration)**
5. **Input Events (Widget, inputEvent)**
6. **Operation Request (Operation, Sensors)**
7. **Operation Binding (Widget, Configuration, inputEvent, Operation, Sensors, ApplID)**
8. **Binding Request (Widget, inputEvent, Operation, Sensors, ApplID)**
10. **Store Operation Binding (Widget, Configuration, inputEvent, Operation, Sensors, ApplID)**
11. **Request (ApplID, Operation, Sensors)**
12. **User Notification (ApplID, Operation, Sensors)**
13. **Data (Sensors)**

G. Petracca et al. - AWARE: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings