CSE543 - Computer and Network Security

Module: Android Security

Professor Trent Jaeger
Classic Computer Systems

Users manually enter information from the outside world
Today’s Computer Systems

Systems measure and sense the physical world
Proof-of-Concept Attacks

Gyrophone: Recognizing Speech From Gyroscope Signals
Yan Michalevsky  Dan Boneh
Computer Science Department
Stanford University
Gabi Nakibly
National Research & Simulation Center
Rafael Ltd.

(sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers
Philip Marquardt
MIT Lincoln Laboratory
244 Wood Street, Lexington, MA USA
philip.marquardt@ll.mit.edu
Arunabh Verma, Henry Carter and
Patrick Traynor
Georgia Institute of Technology
{arunabh.verma@, carterh@, traynor@cc}gatech.edu

ACCessory: Password Inference using Accelerometers on Smartphones
Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, Joy Zhang
{eowusu, junhan, sauvik, perrig, sky}@cmu.edu
Carnegie Mellon University

Inaudible Sound as a Covert Channel in Mobile Devices
Luke Deshotels
North Carolina State University
alecdeshotels@gmail.com

Bridging the Air Gap: Inaudible Data Exfiltration by Insiders
Completed Research Paper
Samuel Joseph O’Malley
University of South Australia
omalsa04@gmail.com
Kim-Kwang Raymond Choo
University of South Australia
raymond.choo@unisa.edu.au

TapLogger: Inferring User Inputs On Smartphone Touchscreens Using On-board Motion Sensors
Zhi Xu
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA, USA
zux103@cse.psu.edu
Kun Bai
IBM T.J. Watson Research Center
Hawthorne, NY, USA
kunbai@us.ibm.com
Sencun Zhu
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA, USA
szhu@cse.psu.edu
Incidents in the Real World

SAN FRANCISCO — Want to invisibly spy on 10 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.

Symantec discovered a new HTTP Android Remote administration tool, named Dendroid, available on the underground market for only $300.

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

Letters Warn Companies of Privacy Risks In Audio Monitoring Technology

FOR RELEASE

March 17, 2016

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

By Katie Dowd, SFGATE  Updated 3:13 pm, Thursday, September 1, 2016

Krysanec trojan: Android backdoor lurking inside legitimate apps

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

$610K Settlement in School Webcam Spy Case

Comments / Share / Tweet / Stumble / Email

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.
Abuse of Sensors

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

Stealthy pictures to recreate a 3D model of a victim’s environment

Demo: https://www.youtube.com/watch?v=ItA791RGvrM

Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones.
Schlegel, Roman and Zhang, Kehuan and Zhou, Xiao-yong and Intwala, Mehool and Kapadia, Apu and Wang, XiaoFeng
Android Permissions

Users grant apps permissions to access sensitive-sensors (cameras, microphones, and screen buffers)

Can we use mandatory access control? What are the challenges?
Permission-Based Systems

Programs can access sensitive-sensors (cameras, microphones, and screen buffers) at any time after the user has authorized them at install time or at first use.

Install-Time

First-Use

Current systems are unable to enforce contextual use of privacy-sensitive sensors.
Abuse of Authorizations

The user grants the camera permission to a augmented reality game (Pokemon GO) to be able to enjoy the gaming experience.
Abuse of Authorizations

The user enjoys playing the game while catching more and more creatures!
The Pokemon GO app opens the camera without the user awareness and captures sensitive frames and send them to a remote server controlled by the adversary.
Abuse of Authorizations

What could we do to address this shortcoming?
Input-Driven Access Control (IDAC)

Every app request for access to a sensitive device must follow a user input event within a retrieved time window

- Does the user know what is the operation associated with the input?
- Does the user know what program is receiving the authorization?

Overhaul: Input-Driven Access Control for Better Privacy on Traditional Operating Systems
K Onarlioglu, W Robertson... - Dependable Systems and ..., 2018 - ieeexplore.ieee.org
User-Driven Access Control (UDAC)

Restricts apps to use trusted gadgets, aka Access Control Gadgets (ACGs) provided by the system

- Does the user know what is the operation associated with the input?
- Does the user know what program is receiving the authorization?
Limitations of Prior Work

How would you attack such defense mechanisms?
GUI Attacks

Previous defenses are subject to **Graphical User Interface attacks**. Programs may leverage the user as **weak point** to get authorizations unwanted by the user.

Users may fail to:

- **Understand the operation** granted by a particular gadget
- **Recognize subtle changes** in the Graphical User Interface (GUI)
- **Identify the application** requesting sensor access
Threat and Trust Model

What are possible threats?

What should be trusted?
**Threat Model**

- Applications (unknown source) can perform any of the UI attacks

**Trust Model**

- Linux kernel and Android OS booted securely (*Secure Boot*)
- *System services* and *system apps* run approved code (shipped with OS)
- SELinux running in *Enforcing Mode*
What security mechanism should be adopted?
Designing Defenses

Objective: Prevent programs from changing the operation associated to a widget arbitrarily

Insights:

• Bind each user input event \( e \) with the widget \( w \) displayed on the screen by the application \( app \)

• Intercept the operation request \( op \) then bind it to the application identity \( app \) and the set of sensors \( S \) targeted by the operation

• Request the user to authorize the operation request explicitly
Designing Defenses

Objective: Prevent programs from changing the operation associated to a widget arbitrarily
Designing Defenses

**Objective:** Prevent programs from changing the operation associated to a widget arbitrarily

**Effect:** Enable the user to verify the association between the operation \( op \) being authorized and the widget \( w \) used to initiate the operation

**Advantages:**

- Avoid authorizing an unwanted operation by a user input event (IDAC)
- Apps are allowed to choose the widgets to associate with particular operations (UDAC)
Designing Defenses

**Objective:** Prevent programs from changing the user interface configuration for a widget

**Insights:**

- Bind the operation request (op) with the user interface configuration (c) used to display the widget (w)
- Define a display context as set of structural features of the most enclosing activity window containing the widget (w)
Designing Defenses

Objective: Prevent programs from changing the user interface configuration for a widget

Effects:

- Identify instance of the same window (i.e., display context) with a different widget
- Identify same widget presented in a different window (i.e., display context)

Advantage: User does not need to check for subtle changes to the widgets or their display context (≠ IDAC and UDAC). Changes detected and flagged by the system automatically.
Designing Defenses

**Objective:** Prevent programs from replacing the foreground activity window of another program

**Insights:**

- Construct an **Activity Window Call Graph** (G) where nodes represent activity windows and edges represent enabled transitions (i.e., user inputs or system events)
- **Security messages** showing app ID and ongoing operation detected and flagged by the system automatically.
Designing Defenses

**Objective:** Prevent programs from replacing the foreground activity window of another program

**Effects:**

- **Activity Window Call Graph** (G) built while the application runs
- Record the relationships among windows used by a program and between programs

**Advantage:** Identify and block activity window hijacking (!= IDAC and UDAC)
What security mechanism does these insights recall?
Operation Binding Concept

**Objective:** make access to privacy-sensitive sensors *explicit* to both the system and the user

System’s View

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

- app = application ID
- op = operation being requested
- S = set of sensors targeted by the request
- e = user input event
- w = user interface widget
- c = user interface configuration containing the widget
  - + activity window call graph (G)

User’s View

![User interface for allowing Instagram to use the front camera](image)
Operation Binding Concept

What security guarantees can it ensure?
**Operation Binding Cache**

**Goal:** maintain a **low authorization effort for users**

**Insights:**
- Use a **caching mechanism** for operation bindings
- Remove operation binding from cache if program changes the way it elicits an operation

**Effect:** “The application will be automatically 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”
Operation Binding Cache

**Goal:** maintain a low authorization effort for users

**Insights:**
- Use a caching mechanism for operation bindings
- Remove operation binding from cache if program changes the way it elicits an operation

**Advantages:**
- Require explicit user's authorization only the first time an operation binding is identified
- Ensure that operation bindings do not become stale
- Prevent an operation from being authorized in multiple ways
- Ensure usability
Experimental Evaluation

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

Research Questions:

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

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

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

(Performance) What is the performance overhead imposed by AWARE for the operation binding construction and enforcement?
Protection - Laboratory-Based User Study

**SimpleFilters** test malware app:

- provides filtering features pictures and videos
- performs UI attacks accessing camera and microphone to collect sensitive information
## Experimental Evaluation

### Protection - Laboratory-Based User Study

(90 participants - 6 groups)

<table>
<thead>
<tr>
<th>Task Description (Randomized)</th>
<th>Attack Scenario</th>
<th>Authorization Requests (△ AWARE)</th>
<th>Attack Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TASK 1</strong>  : Take a picture with the smartphone’s front camera by using the SimpleFilters app.</td>
<td>Operation Switching: The SimpleFilters app also starts recording audio via the smartphone’s microphone instead of only taking a picture.</td>
<td>• Allow SimpleFilters to use the Front Camera and Microphone to Record Video when pressing ⌘ ?</td>
<td>Group1 (Install-Time): 100% Group2 (First-Use): 93% Group3 (Input-Driven): 100% Group4 (Developer ID): 100% Group5 (AC Gadgets): 0% Group6 (AWARE): 0%</td>
</tr>
<tr>
<td><strong>TASK 2</strong>  : Take a picture with the front camera by using the SimpleFilters app.</td>
<td>Bait-and-Context-Switch: We make the video camera widget appear in the photo capture window, with a camera preview, to trick the user into allowing SimpleFilters to record audio instead of just take a picture.</td>
<td>• Allow SimpleFilters to use the Front Camera and Microphone to Record Video when pressing ⌘ ?</td>
<td>Group1 (Install-Time): 87% Group2 (First-Use): 87% Group3 (Input-Driven): 93% Group4 (Developer ID): 87% Group5 (AC Gadgets): 87% Group6 (AWARE): 7%</td>
</tr>
<tr>
<td><strong>TASK 3</strong>  : Take six consecutive pictures with the smartphone’s front camera by using the SimpleFilters app.</td>
<td>Bait-and-Widget-Switch: Before the participants took the fifth picture, the SimpleFilters app replaced the camera widget with the video camera widget to enable video recording instead. The camera button was restored before the users took the sixth picture.</td>
<td>• Allow SimpleFilters to use the Front Camera and Microphone to record Video when pressing ⌘ ?</td>
<td>Group1 (Install-Time): 87% Group2 (First-Use): 87% Group3 (Input-Driven): 93% Group4 (Developer ID): 87% Group5 (AC Gadgets): 87% Group6 (AWARE): 7%</td>
</tr>
<tr>
<td><strong>TASK 4</strong>  : Record a voice note using the Keep app.</td>
<td>Identity Spoofing: We let the participants select the Keep app from the app menu, however, we programmatically triggered the SimpleFilters app to hijack the on-screen activity and spoof the Keep app.</td>
<td>• Allow SimpleFilters to use the Microphone to Record Audio when pressing ⌘ ?</td>
<td>Group1 (Install-Time): 93% Group2 (First-Use): 93% Group3 (Input-Driven): 93% Group4 (Developer ID): 47% Group5 (AC Gadgets): 93% Group6 (AWARE): 0%</td>
</tr>
</tbody>
</table>
Experimental Evaluation

Usability: Field-Based User Study (24 participants)

21 popular apps (7 categories)*
Experiment duration: 1 week

*Average number of apps installed by users in personal smartphones (source: www.statistica.com)

Experimental results:
• On average 2 additional explicit authorizations per application
• 4 apps - Number of explicit authorization equal to first-use
• Higher but limited number of explicit authorizations (at most 12)

<table>
<thead>
<tr>
<th>App Category</th>
<th>App Name</th>
<th>Explicit User Authorizations First-Use</th>
<th>Average</th>
<th>Total Operation Authorizations Avg. (s.d.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Recording</td>
<td>WhatsApp</td>
<td>3</td>
<td>6 (±1)</td>
<td>1,217 (±187)</td>
</tr>
<tr>
<td></td>
<td>Video Messenger</td>
<td>3</td>
<td>1 (±0)</td>
<td>889 (±99)</td>
</tr>
<tr>
<td></td>
<td>Facebook</td>
<td>2</td>
<td>4 (±1)</td>
<td>3,864 (±223)</td>
</tr>
<tr>
<td></td>
<td>SilentEye</td>
<td>2</td>
<td>5 (±1)</td>
<td>234 (±20)</td>
</tr>
<tr>
<td></td>
<td>Fideo</td>
<td>2</td>
<td>4 (±1)</td>
<td>213 (±23)</td>
</tr>
<tr>
<td>Screenshot Capture</td>
<td>Ok Screenshot</td>
<td>1</td>
<td>2 (±1)</td>
<td>49 (±8)</td>
</tr>
<tr>
<td></td>
<td>Screenshot Easy</td>
<td>1</td>
<td>2 (±1)</td>
<td>76 (±7)</td>
</tr>
<tr>
<td></td>
<td>Screenshot Capture</td>
<td>1</td>
<td>2 (±1)</td>
<td>64 (±4)</td>
</tr>
<tr>
<td>Screen Recording</td>
<td>REC Screen Recorder</td>
<td>2</td>
<td>3 (±1)</td>
<td>41 (±8)</td>
</tr>
<tr>
<td></td>
<td>AZ Screen Recorder</td>
<td>2</td>
<td>4 (±2)</td>
<td>49 (±7)</td>
</tr>
<tr>
<td></td>
<td>Rec.</td>
<td>2</td>
<td>3 (±1)</td>
<td>66 (±4)</td>
</tr>
<tr>
<td>Full Screen Mode</td>
<td>Instagram</td>
<td>2</td>
<td>6 (±1)</td>
<td>3,412 (±182)</td>
</tr>
<tr>
<td></td>
<td>Snapchat</td>
<td>2</td>
<td>6 (±1)</td>
<td>5,287 (±334)</td>
</tr>
<tr>
<td></td>
<td>Skype</td>
<td>2</td>
<td>9 (±3)</td>
<td>468 (±62)</td>
</tr>
<tr>
<td>Remote Control</td>
<td>Prey Anti Theft</td>
<td>2</td>
<td>8 (±2)</td>
<td>47 (±5)</td>
</tr>
<tr>
<td></td>
<td>Lost Android</td>
<td>2</td>
<td>6 (±1)</td>
<td>37 (±6)</td>
</tr>
<tr>
<td></td>
<td>Avast Anti-Theft</td>
<td>2</td>
<td>4 (±1)</td>
<td>34 (±7)</td>
</tr>
<tr>
<td>Hands-Free Control</td>
<td>Google Voice Search</td>
<td>1</td>
<td>1 (±0)</td>
<td>1,245 (±122)</td>
</tr>
<tr>
<td></td>
<td>HappyShutter</td>
<td>1</td>
<td>1 (±0)</td>
<td>3 (±1)</td>
</tr>
<tr>
<td></td>
<td>SnapClap</td>
<td>1</td>
<td>1 (±0)</td>
<td>4 (±2)</td>
</tr>
</tbody>
</table>
Compatibility:

Android Compatibility Test Suite

- 1,000 most-downloaded apps from Google Play
- 13 hours and 28 minutes
- 126,681 passed tests over 126,686

Only five minor compatibility issues addressed in subsequent prototypes: [Viber] Camera and microphone probing at reboot (No impact on video or voice calls)
Experimental Evaluation

Performance: Android UI/Application Exerciser
(1,000 most-downloaded apps from Google Play)

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

Experimental results:

- 0.33% system-wide overhead
- Order of tens of microseconds per access (unnoticeable to users)
- 3 MB of cache (operation bindings)