26TH USENIX
SECURITY SYMPOSIUM

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

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

Trent Jaeger
tjaeger@cse.psu.edu
The Pennsylvania State University
School of Electrical Engineering and Computer Science
Institute for Networking and Security Research

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

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

UNIVERSITY OF OXFORD

PennState College of Engineering

Technische Universität München

Increasing Availability of Privacy-Sensitive Sensors

PennState

Controlling when **applications** may use **privacy-sensitive sensors** (i.e., cameras, microphones, and touch screens):







Entertainment

Screen Sharing

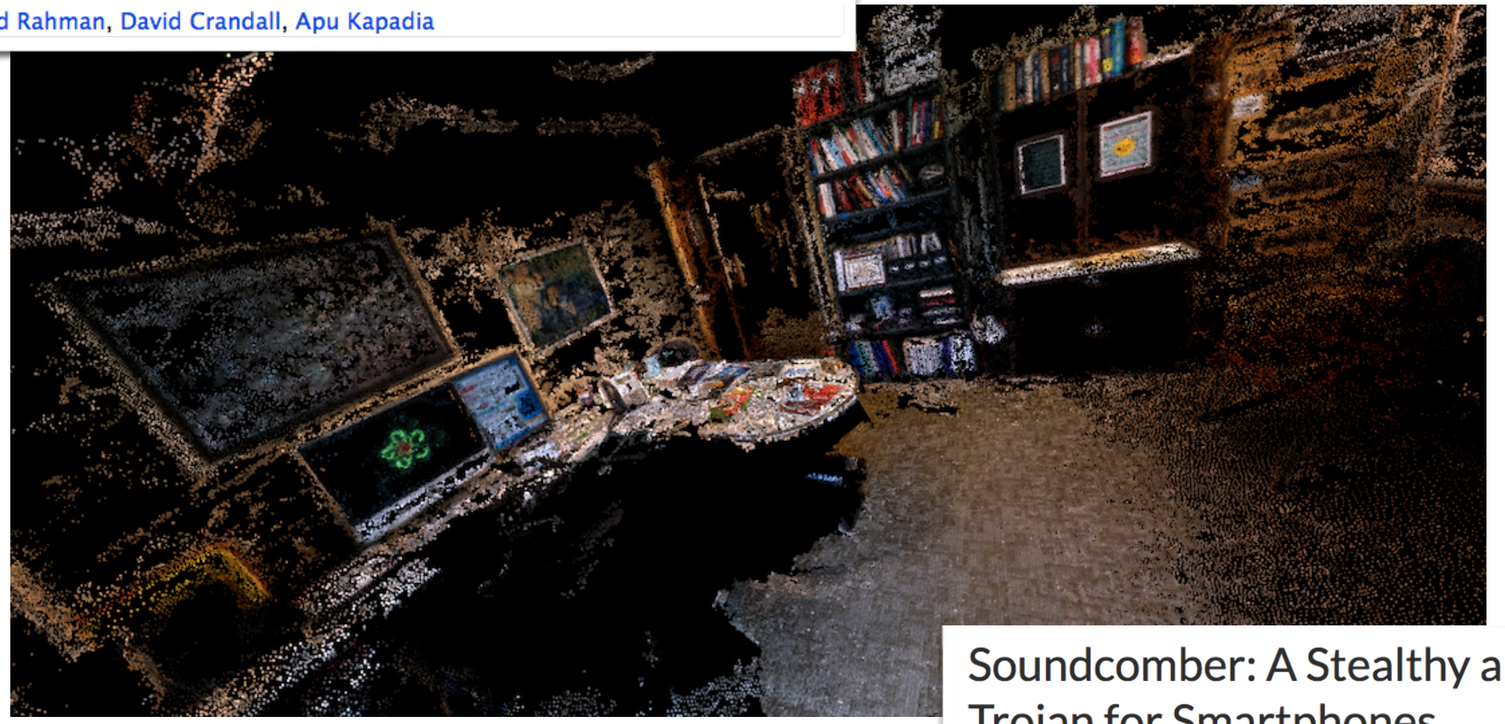
Banking

Abuse of Privacy-Sensitive Sensors

PennState

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, Kehuan and Zhou, Xiao-yong and Intwala, Mehool and Kapadia, Apu and Wang, XiaoFeng

Real World Incidents

PennState

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

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.

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

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.

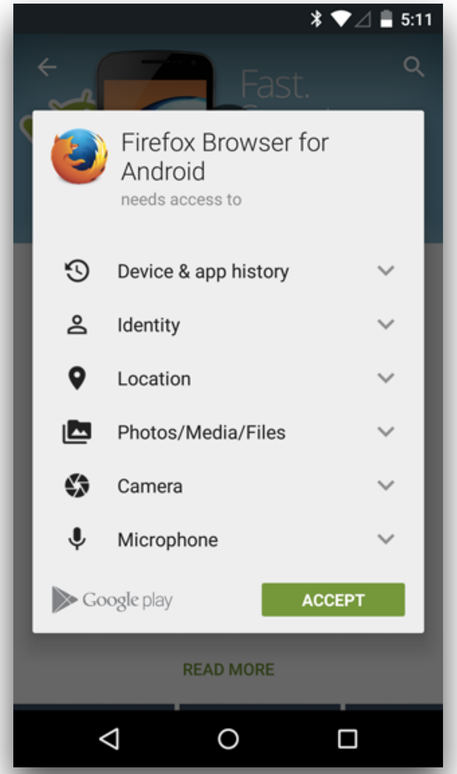
FTC Issues Warning Letters to App Developers Using 'Silverpush' Code

Letters Warn Companies of Privacy Risks In Audio Monitoring Technology

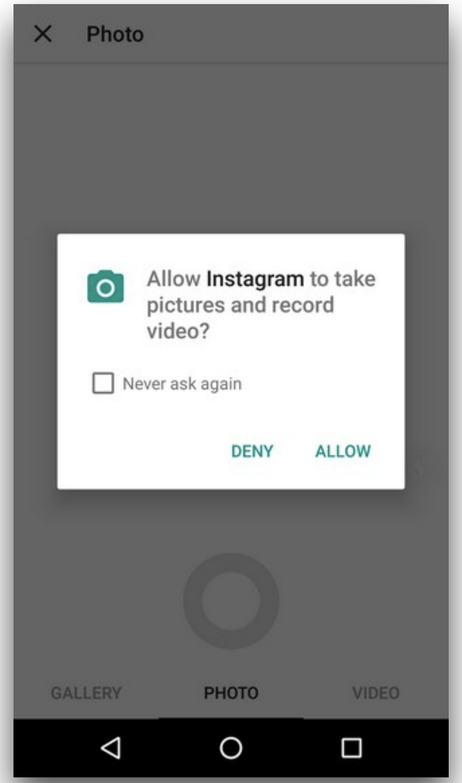
Current Authorization Mechanisms

PennState

Install-Time



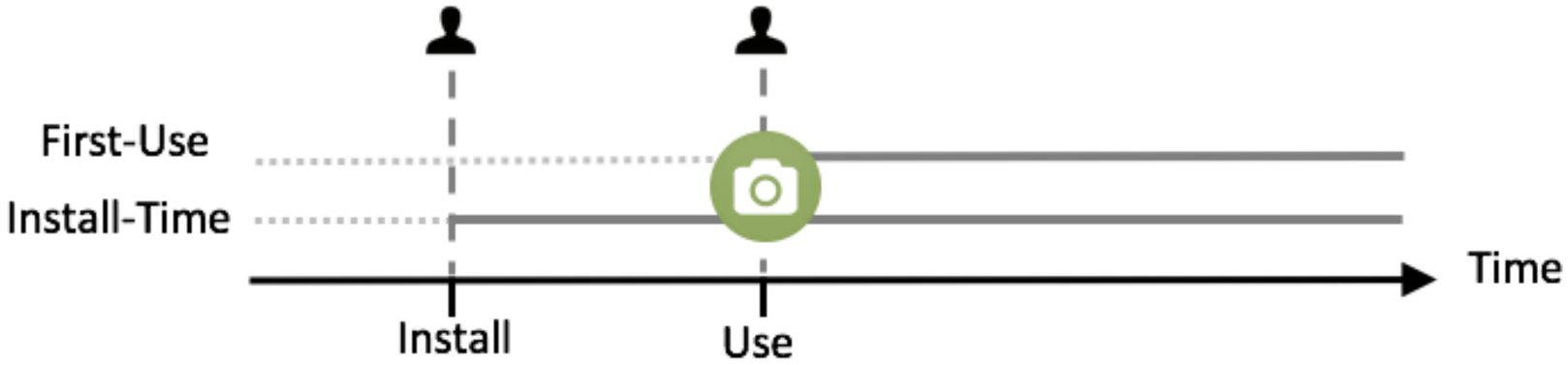
First-Use



Beginning in Android 6.0 (API level 23), users grant permissions to apps while the app is running, not whey the install the app!

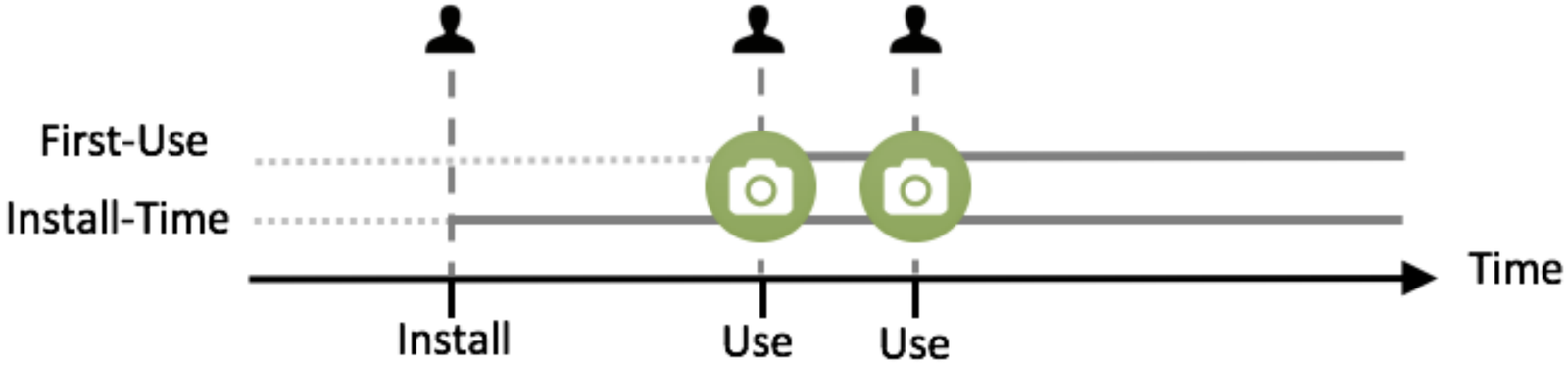
Shortcomings



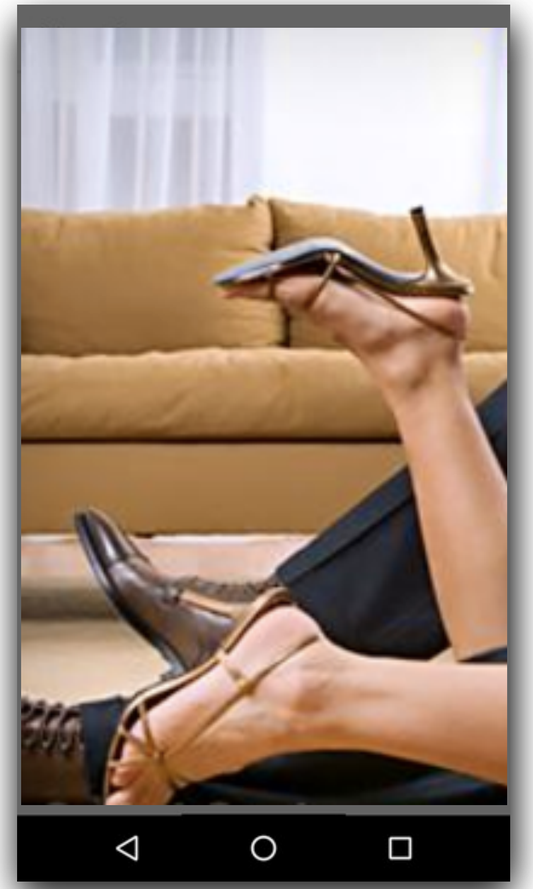


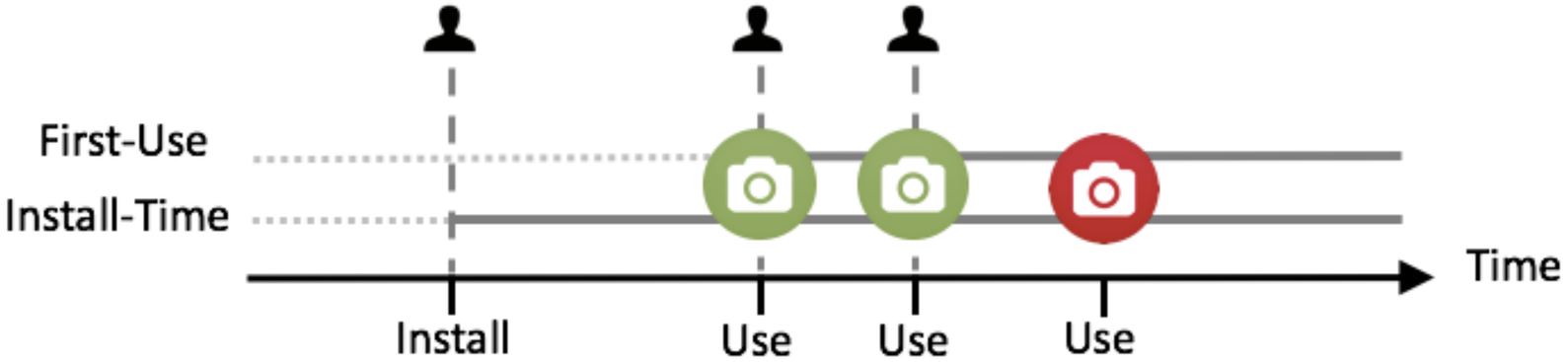
Shortcomings





Shortcomings



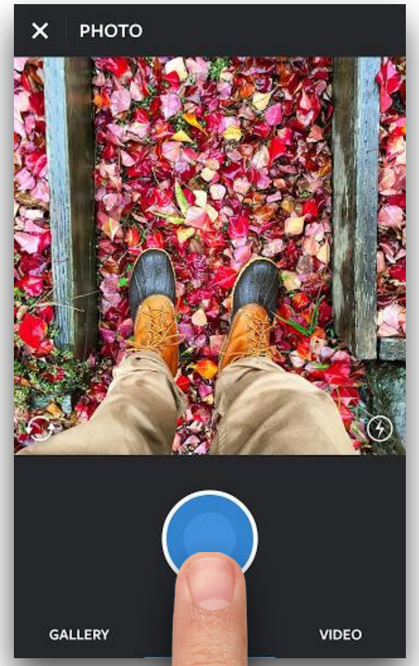


Proposed Defenses

PennState

Input-Driven Access Control (IDAC)

Authorize an operation request that immediately follows a user input event



Overhaul: **Input-Driven** Access Control for Better Privacy on Traditional Operating Systems
K Onarlioglu, W Robertson… - Dependable Systems and …, 2016 - ieeexplore.ieee.org

User inputs associated with operation authorizations

Binding between the user inputs and the authorized operations still unknown to the system!

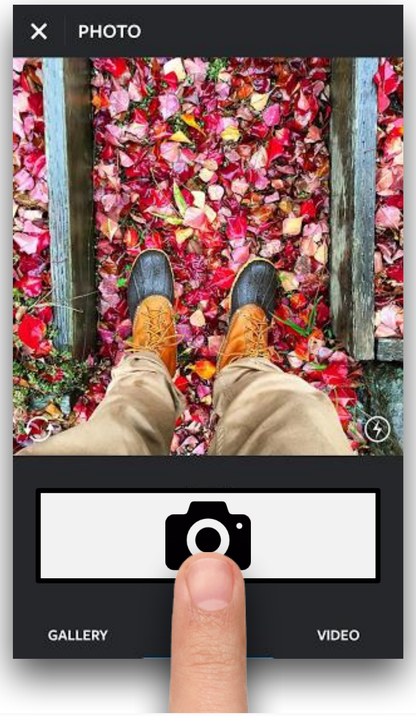
Proposed Defenses

PennState

User-Driven Access Control (UDAC)

Applications must use **system-defined gadgets**

associated with particular operations



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

User-driven access control: Rethinking permission granting in modern operating systems
F Roesner, T Kohno, A Moshchuk... - Security and privacy ..., 2012 - ieeexplore.ieee.org

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

Binding still not explicit to the user!

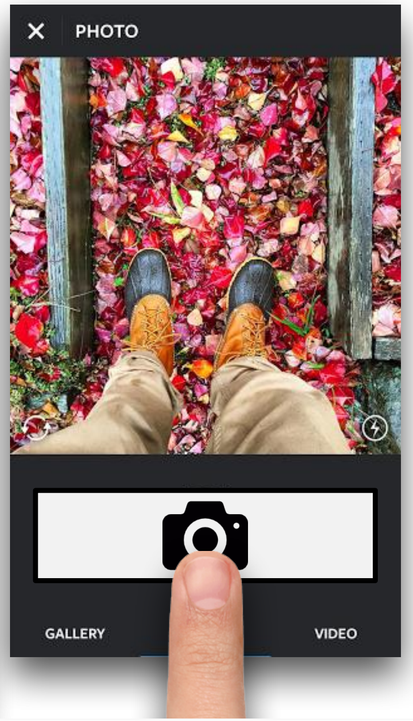
Proposed Defenses

User-Driven Access Control (UDAC)

Applications must use system-defined gadgets associated with particular operations

Compatibility Issue





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

User-driven access control: Rethinking permission granting in modern operating systems
F Roesner, T Kohno, A Moshchuk… - Security and privacy …, 2012 - ieeexplore.ieee.org

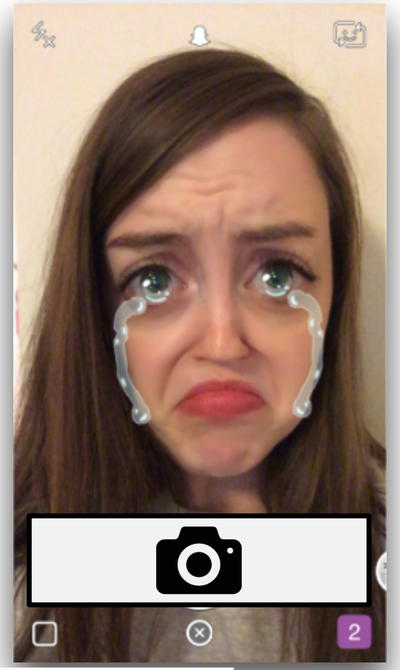
Proposed Defenses

User-Driven Access Control (UDAC)

Applications must use system-defined gadgets associated with particular operations

3,000,000+ apps
Need Redesign





No Customization

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

User-driven access control: Rethinking permission granting in modern operating systems
F Roesner, T Kohno, A Moshchuk… - Security and privacy …, 2012 - ieeexplore.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)

PennState

Interaction #1



Use Interface Attacks (Bait-and-Switch)

PennState

Interaction #2



Use Interface Attacks (Bait-and-Switch)

PennState

Interaction #3



Use Interface Attacks (Bait-and-Switch)

PennState

Interaction #4



Use Interface Attacks (Bait-and-Switch)

Interaction #5



Use Interface Attacks (Bait-and-Switch)

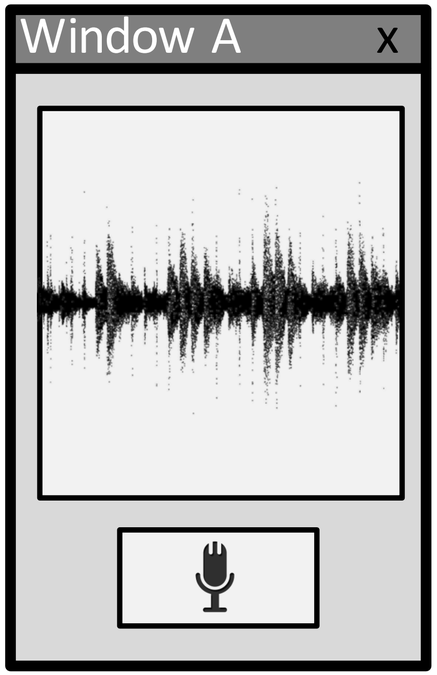
Interaction #4



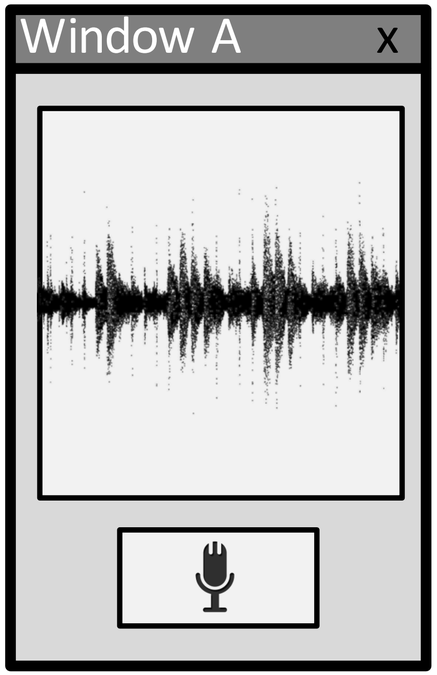
The application maintained the windowing display context but switched the widget to **record audio**

“Bait-and-Switch Attack”

Use Interface Attacks (Application Spoofing)



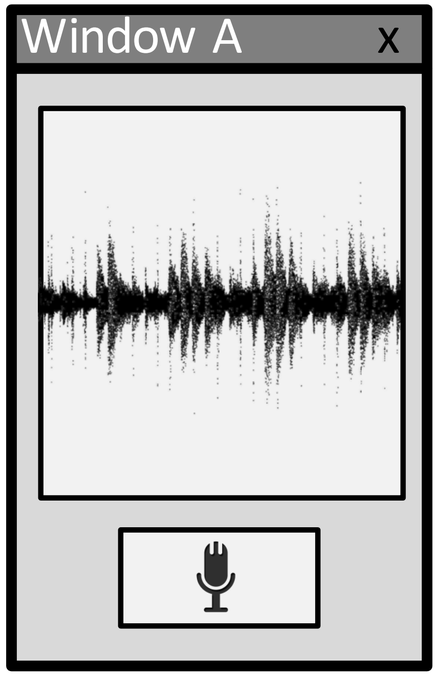
Use Interface Attacks (Application Spoofing)



Google

A click by the user allows
the **Legitimate App**
to record audio

Use Interface Attacks (Application Spoofing)



MR. ROBOT

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

"Application Spoofing Attack"

Research Objectives

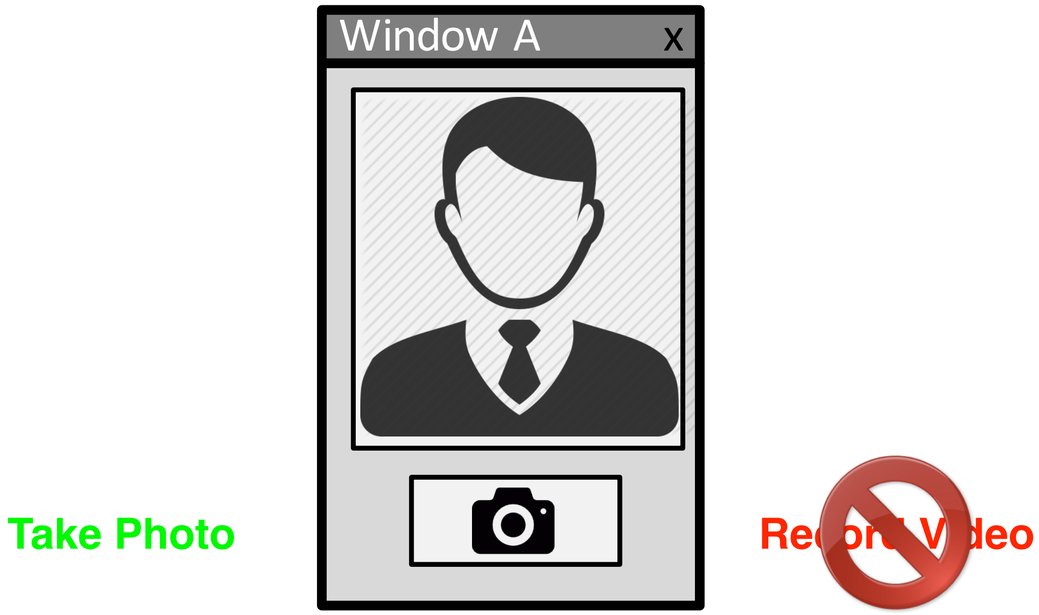
- Prevent User Interface Attacks
- Maintain a low authorization effort for the user
- Ensure compatibility with existing applications
- Ensure a performance overhead not perceivable by the user



Preventing Operation Switching Attacks

PennState

Goal: Prevent applications from changing the operation associated to a widget arbitrarily

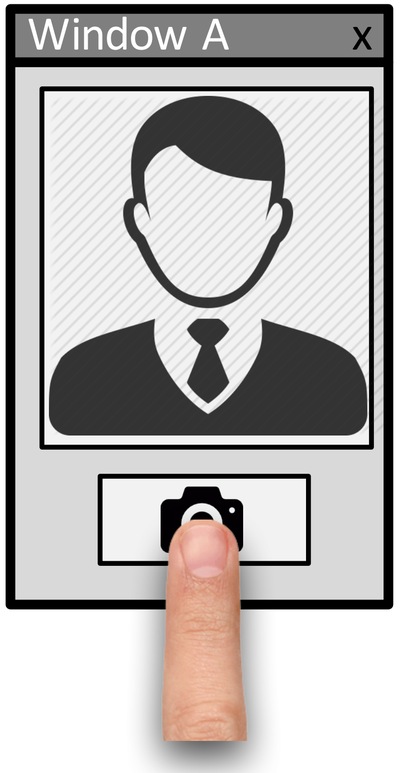


Preventing Operation Switching Attacks

Goal: Prevent applications 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**)



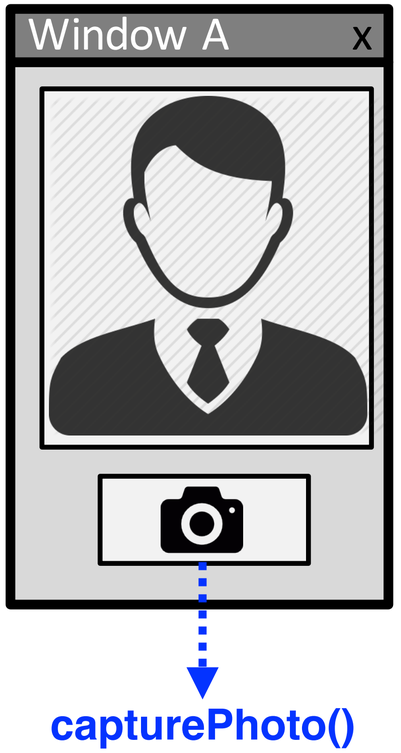
Preventing Operation Switching Attacks

PennState

Goal: Prevent applications 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



Preventing Operation Switching Attacks

PennState

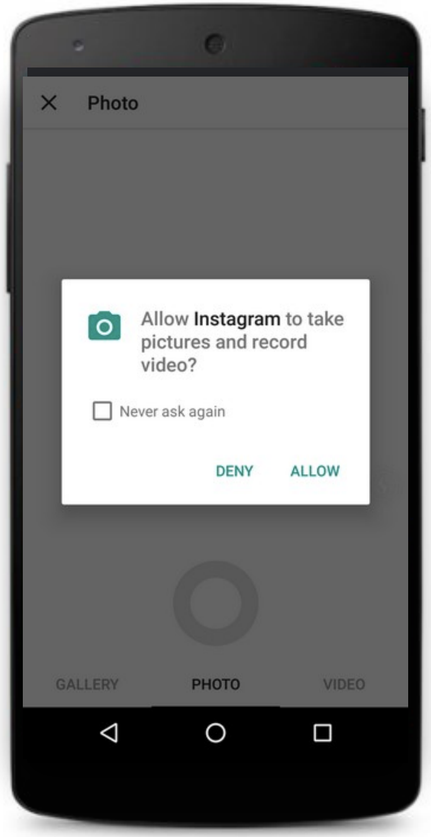
Goal: Prevent applications from changing the operation associated to a widget arbitrarily

Insights:

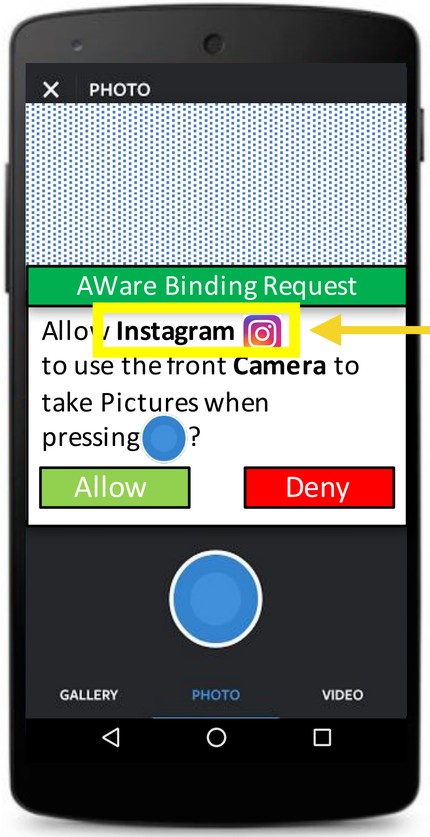
- Bind each user input event (**e**) with the widget (**w**) displayed on the screen by the application (**app**)
- Intercept the operation request (**op**) then bind it to the application identity (app) and the set of sensors (**S**) targeted by the operation
- Request the user to **authorize the binding request explicitly**



AWARE's Explicit Binding Request



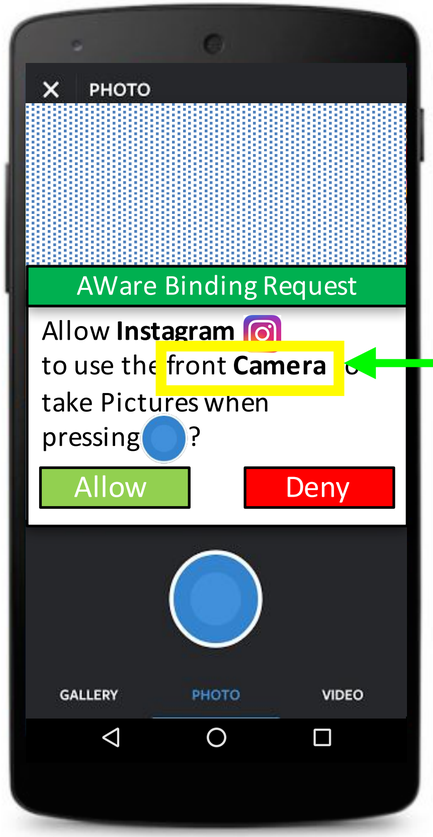
Currently
(First-Use)



app (Application ID)

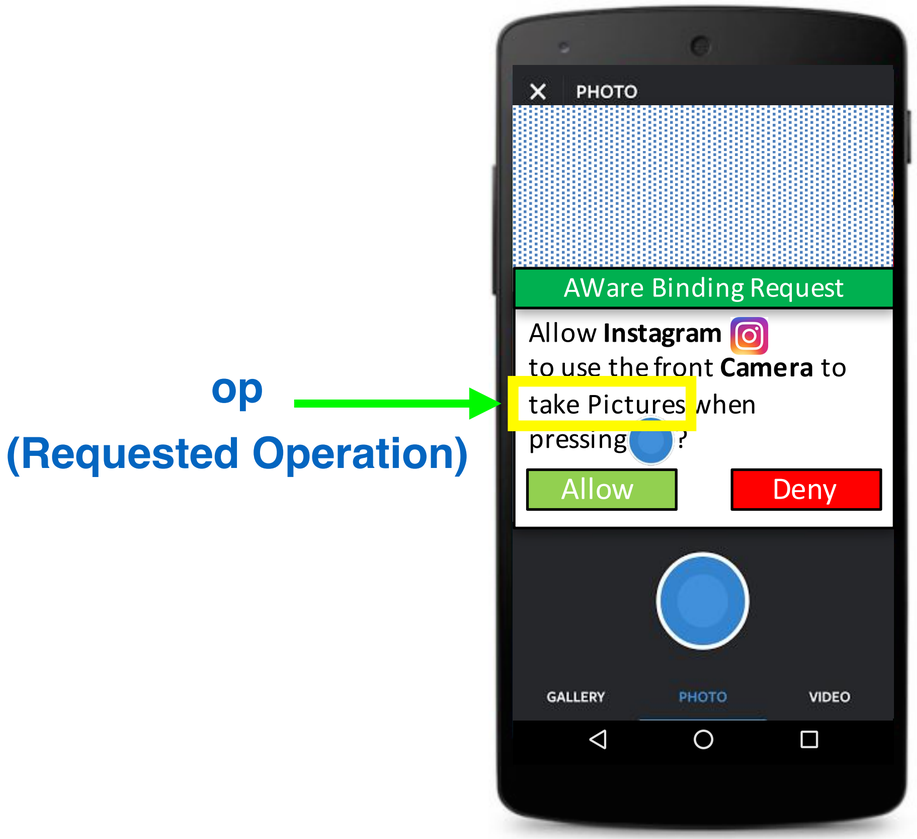
AWARE

AWARE's Explicit Binding Request



S (Set of Sensors)

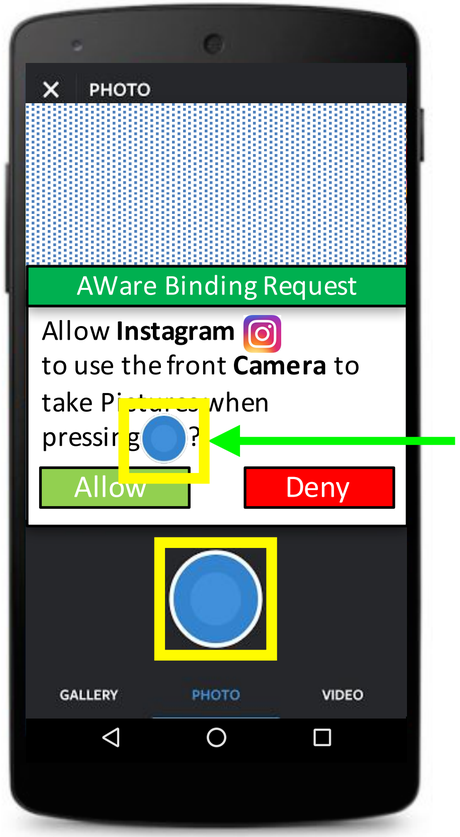
AWARE's Explicit Binding Request



AWARE's Explicit Binding Request



AWARE's Explicit Binding Request



w (Widget)

Preventing Operation Switching Attacks

Goal: Prevent applications from changing the operation associated to a widget arbitrarily

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

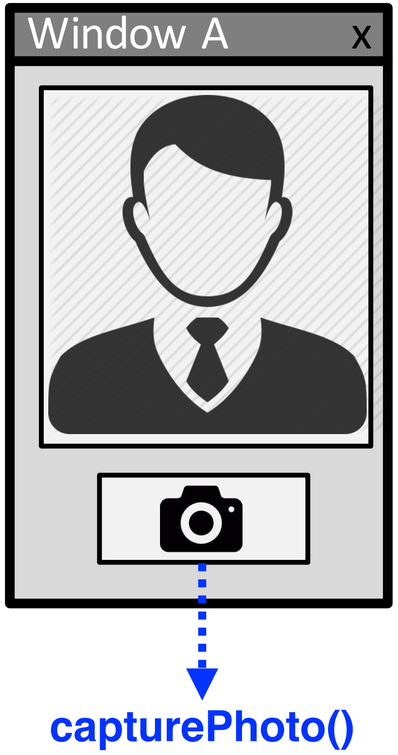


Advantages:

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

Preventing Bait-and-Switch Attacks

Goal: Prevent applications from changing the user interface configuration for a widget

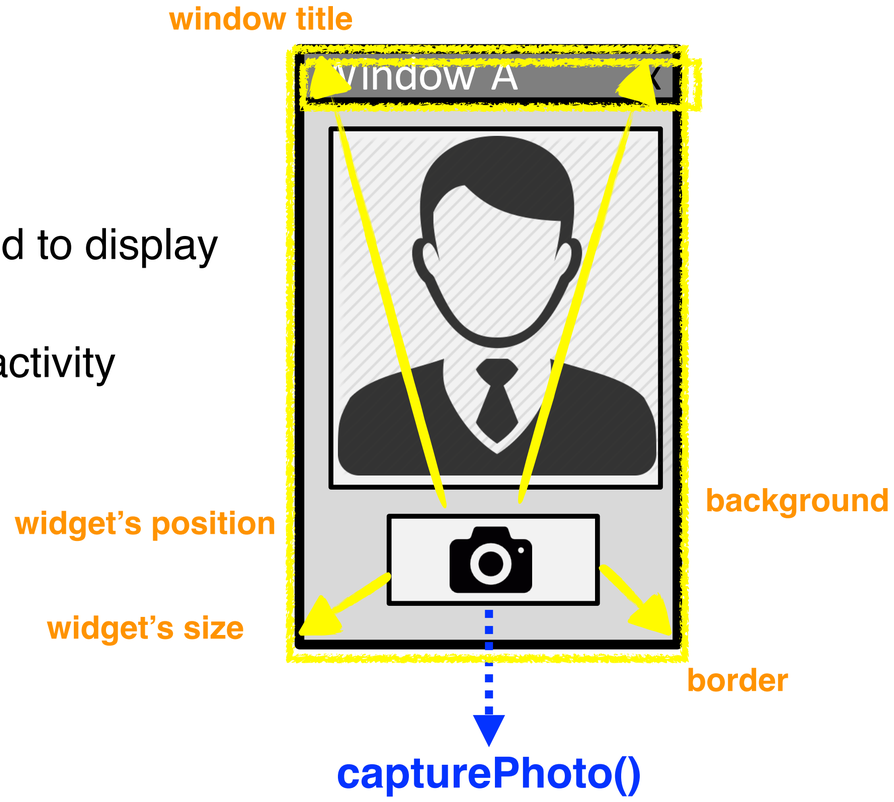


Preventing Bait-and-Switch Attacks

Goal: Prevent applications from changing the user interface configuration for a widget

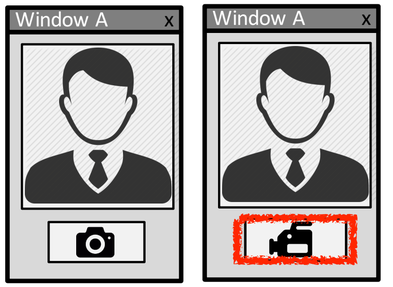
Insights:

- 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**)



Preventing Bait-and-Switch Attacks

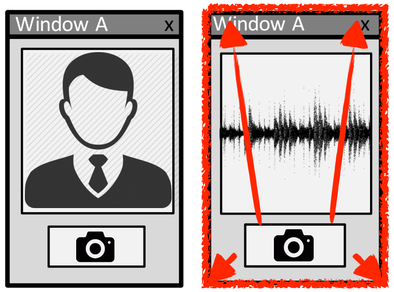
Goal: Prevent applications from changing the user interface configuration for a widget



Effects:

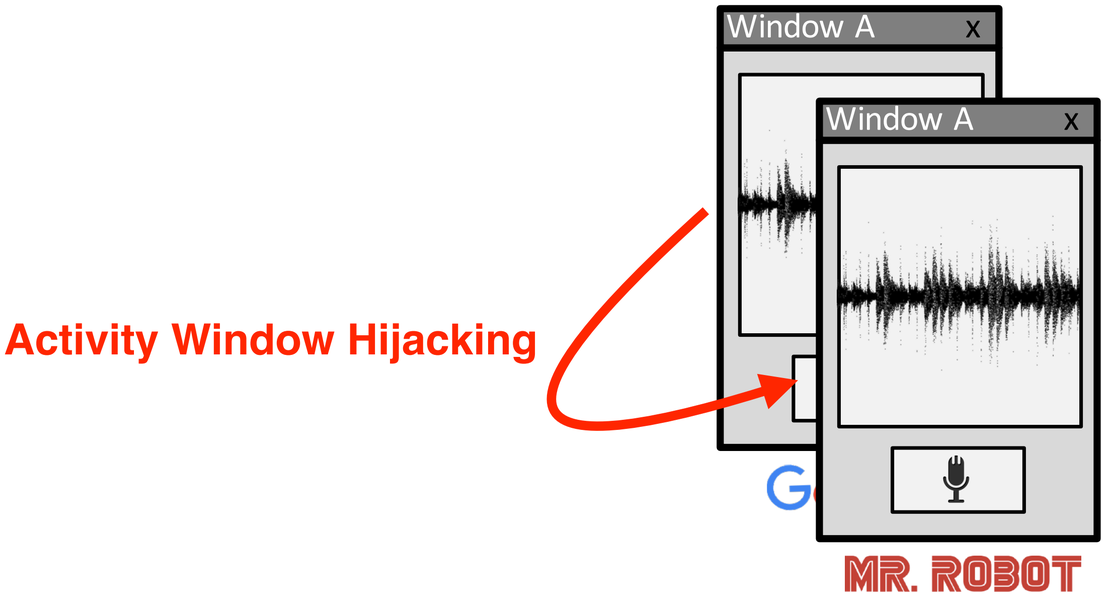
- Identify instance of the same window (i.e., display context) with a **different widget**
- Identify same widget presented in a **different window** (i.e., display context)

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.



Preventing Application Spoofing Attacks

Goal: Prevent applications from replacing the foreground activity window of another application

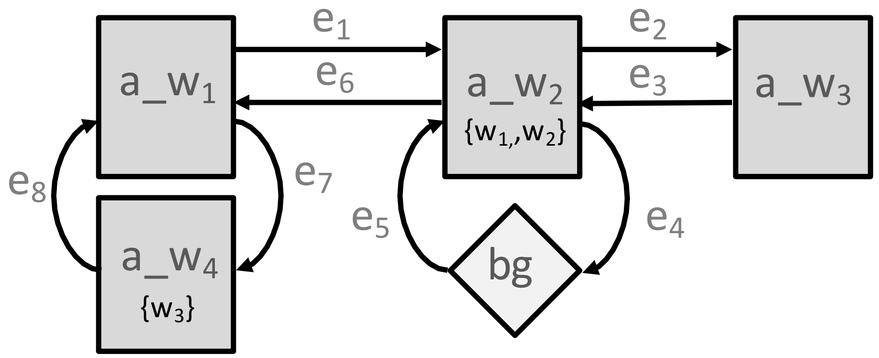


Preventing Application Spoofing Attacks

PennState

Goal: Prevent applications from replacing the foreground activity window of another application

Insight: Construct an **Activity Window Call Graph** (**G**) where nodes represent activity windows and edges represent enabled transitions (i.e., user inputs or system events)



Preventing Application Spoofing Attacks

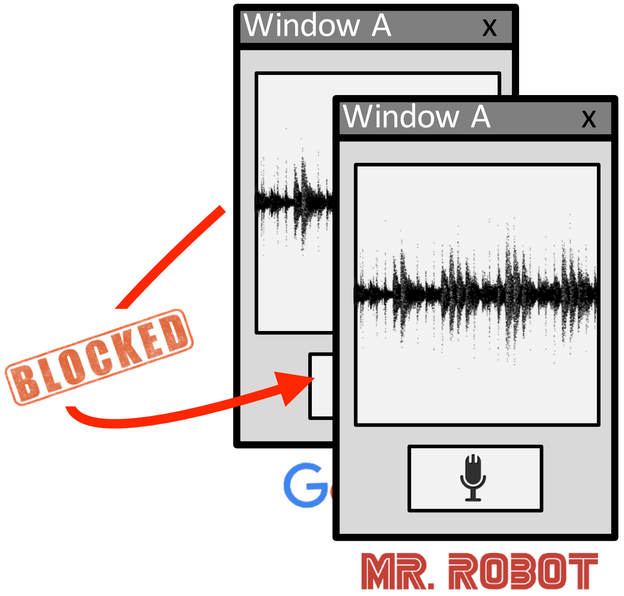
PennState

Goal: Prevent applications from replacing the foreground activity window of another application

Effects:

- Activity Window Call Graph (G) built while the application runs
- Record the relationships among windows used by an application

Advantage: Identify and block activity window hijacking (IDAC and UDAC)



AWARE's Operation Binding

AWARE: Authorization Framework extending OS middleware
to make access to privacy-sensitive sensors **explicit** to both the system and the user

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

app = application associated with widget and operation request

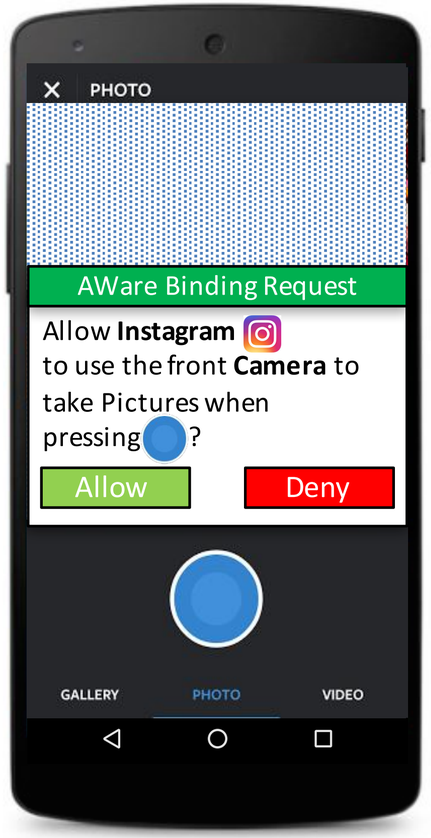
S = set of sensors targeted by the request

op = operation being requested

e = user input event

w = user interface widget

c = user interface configuration containing the widget + activity window call graph



System **User**

Maintain a Low Authorization Effort for the User

Goal: Limit the number of explicit authorizations by the user

Insights:

- Use a **caching mechanism** for operation bindings
- Remove an operation binding from cache if an app 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"

Advantages:

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

Ensure Compatibility with Existing Apps

Goal: Allow applications to choose how they elicit user approval for use of a sensor

Insights:

- No external libraries
- No code annotation
- No app code rewriting
- Dynamic monitoring and creation of operation bindings

Effect: Can be integrated with existing off-the-shelf operating systems

Advantages:

- Facilitate adoption and deployability
- No effort or burden for app developers



Experimental Evaluation



Prototyped (Android OS 6.0.1_r5)

Tested (Nexus 5 and Nexus 5X smartphones)

Research Questions:

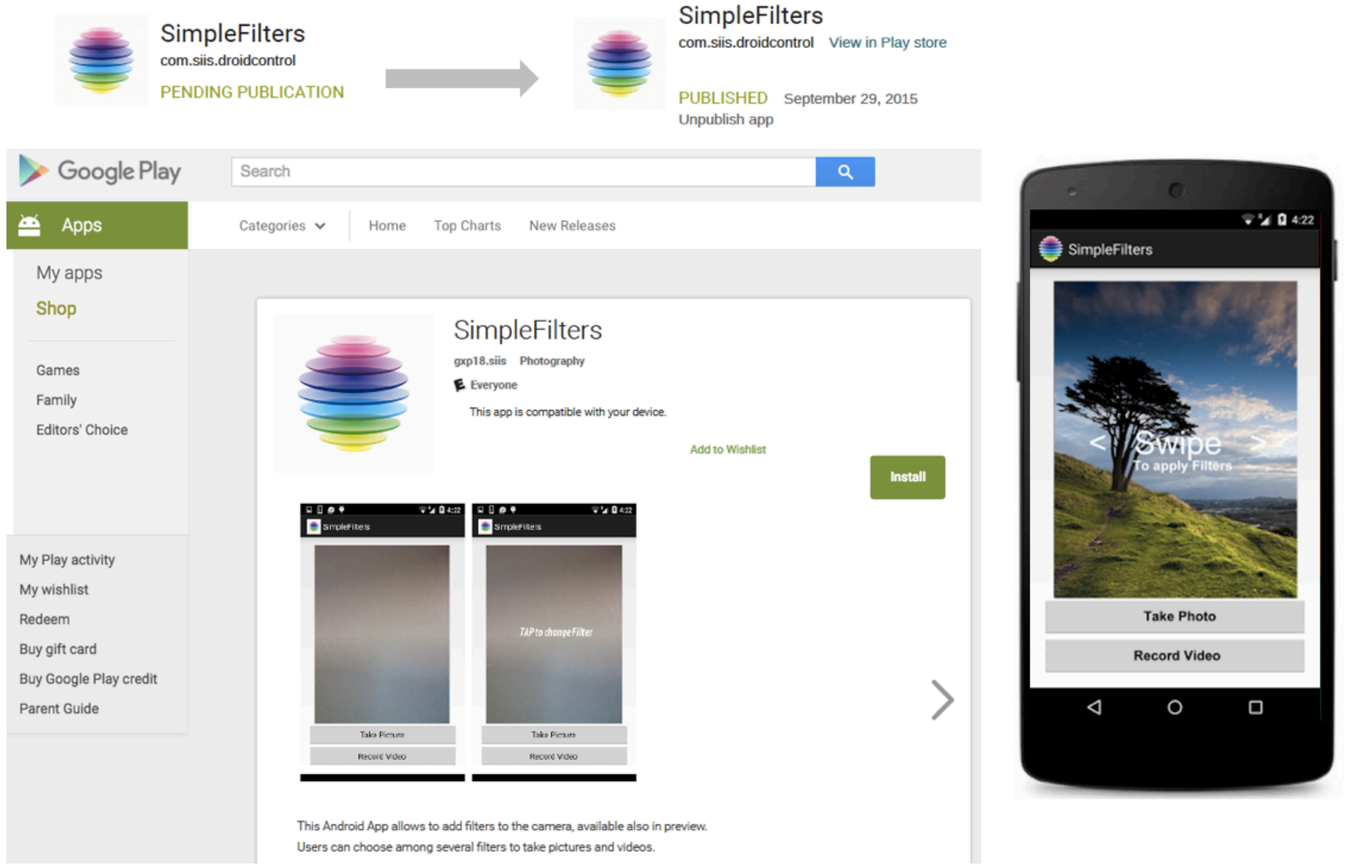
- *To what degree is the **AWARE** operation binding concept assisting the users in avoiding attacks?* (**Effectiveness**)
- *What is the decision overhead imposed to users due to per-configuration access control?* (**Usability**)
- *How many existing apps malfunctioned due to the integration of **AWARE**?* (**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**



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**

Task Description (Randomized)	Attack Scenario
TASK 1 : Take a picture with the smartphone's front camera by using the `SimpleFilters` app.	Operation Switching: The `SimpleFilters` app also starts recording audio via the smartphone's microphone instead of only taking a picture.
TASK 2 : Take a picture with the front camera by using the `SimpleFilters` app.	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. □
TASK 3 : Take six consecutive pictures with the smartphone's front camera by using the `SimpleFilters` app.	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. □
TASK 4 : Record a voice note using the `Keep` app.	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.

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%**

Task Description (Randomized)	Attack Scenario
TASK 1 : Take a picture with the smartphone's front camera by using the `SimpleFilters` app.	Operation Switching: The `SimpleFilters` app also starts recording audio via the smartphone's microphone instead of only taking a picture.
TASK 2 : Take a picture with the front camera by using the `SimpleFilters` app.	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. □
TASK 3 : Take six consecutive pictures with the smartphone's front camera by using the `SimpleFilters` app.	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. □
TASK 4 : Record a voice note using the `Keep` app.	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.

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

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

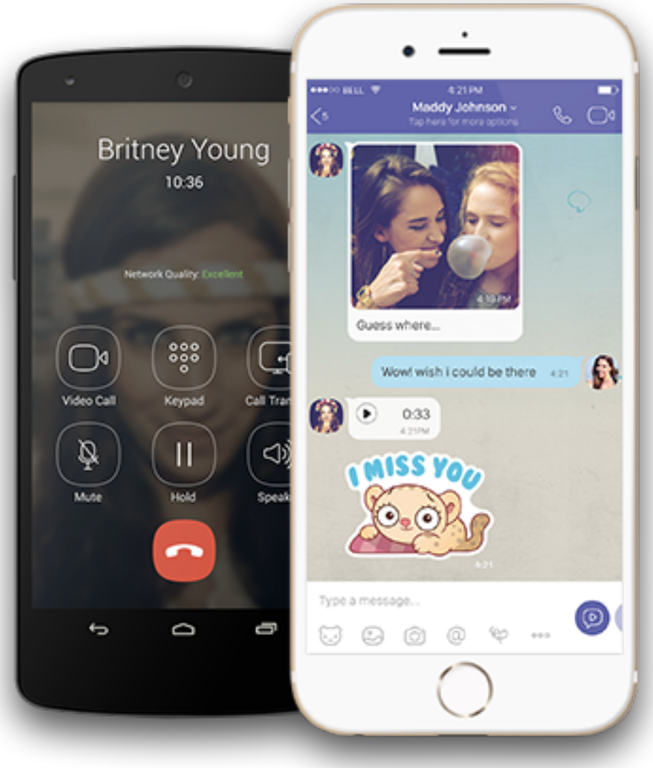
*www.statistica.com

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)

Performance

*What is the **performance overhead** imposed by **AWARE** for the operation binding construction and enforcement?*

Android UI/Application Exerciser Monkey:

- 1,000 apps (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 memory** for the operation binding cache and window call graphs

Conclusion

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

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

- **Compatible with 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

UNIVERSITY OF OXFORD

PennState College of Engineering

Technische Universität München

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

Research Funded by:

ARL

Approach Overview

PennState

AWARE Authorization Workflow

